helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

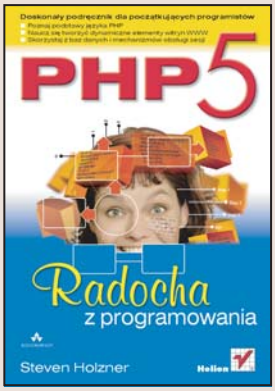
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

PHP5. Radocha z programowania



Autor: Steven Holzner
Tłumaczenie: Radosław Meryk
ISBN: 83-246-0085-X
Tytuł oryginału: Spring Into PHP 5
Format: B5, stron: 344

Doskonały podręcznik dla początkujących programistów

- Poznaj podstawy języka PHP
- Naucz się tworzyć dynamiczne elementy witryn WWW
- Skorzystaj z baz danych i mechanizmów obsługi sesji

Popularność języka PHP ciągle rośnie. Twórcom i administratorom witryn WWW nie wystarcza już standardowy HTML – potrzebują narzędzia pozwalającego na kontrolę odwiedzin witryny, łatwą edycję artykułów, pobieranie i przesyłanie plików oraz obsługę danych przekazywanych z formularzy. Wszystkie te możliwości oferuje PHP i witryny WWW wykonane w tej technologii. Najnowsza wersja języka – PHP 5, to w pełni obiektowy język programowania, pozwalający na tworzenie rozbudowanych aplikacji WWW, nad którymi zarówno twórcy, jak i użytkownicy mogą mieć pełną kontrolę.

„PHP5. Radocha z programowania" to podręcznik, dzięki któremu szybko poznasz język PHP i napiszesz swoje pierwsze aplikacje WWW. Czytając tę książkę, dowiesz się, z jakich podstawowych elementów składa się program w języku PHP, jak łączyć kod PHP ze znacznikami HTML i sterować przebiegiem programu. Nauczysz się tworzyć skrypty weryfikujące i przetwarzające dane z formularzy, łączące się z bazami danych i wysyłające wiadomości e-mail. Każde z zagadnień poznasz w oparciu o praktyczne przykłady, które z łatwością dostosujesz do swoich wymagań.

- Instalacja PHP
- Łączenie PHP i HTML
- Zmienne, operatory i instrukcje
- Przetwarzanie ciągów znaków
- Tworzenie i wykorzystywanie funkcji
- Obsługa formularzy na stronach WWW
- Programowanie obiektowe w PHP5
- Komunikacja z bazami danych
- Obsługa sesji i plików cookie
- Przesyłanie plików za pomocą protokołu FTP

Przekonaj się, jak łatwe jest programowanie w PHP5

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

Spis treści

O autorze **9**

Przedmowa **11**

Rozdział 1. Podstawowe wiadomości o PHP **15**

Jak uzyskać dostęp do PHP? 16
Konfiguracja środowiska programistycznego 18
Pierwszy skrypt PHP 20
Uruchomienie pierwszego skryptu PHP 22
Co zrobić, jeśli nie zadziała? 22
Łączenie kodu HTML i PHP 24
Wyświetlanie tekstu 25
Dodatkowe możliwości wyświetlania tekstów 27
Wyświetlanie całych dokumentów 29
Uruchamianie skryptów PHP z wiersza polecenia 30
Zastosowanie komentarzy 32
Uchwyty do danych: zmienne 34
Przypisywanie wartości do zmiennych 35
Interpolacja zmiennych w ciągach znaków 37
Tworzenie zmiennych zawierających nazwy zmiennych 39
Stałe 41
Typy danych 43
Podsumowanie 45

Rozdział 2. Operatory i sterowanie przepływem **47**

Operatory arytmetyczne 48
Funkcje arytmetyczne 48
Operatory przypisania 51
Inkrementacja i dekrementacja 52
Priorytety operatorów 53
Operator wykonania 55
Operatory znakowe 56
Operatory bitowe 57
Zastosowanie instrukcji if 58
Operatory porównań 60
Operatory logiczne 62
Zastosowanie instrukcji else 63
Zastosowanie instrukcji elseif 64
Operator trójskładnikowy 65

Zastosowanie instrukcji switch ... 66
Pętle for ... 67
Pętle while ... 69
Pętle do...while ... 71
Pętle foreach ... 72
Przerwanie działania pętli ... 73
Zastosowanie instrukcji continue ... 74
Składnia alternatywna ... 75
Podsumowanie ... 76

Rozdział 3. Przetwarzanie ciągów znaków i tablic ... 77
Funkcje obsługi ciągów znaków ... 78
Zastosowanie funkcji obsługi ciągów znaków ... 80
Formatowanie ciągów znaków ... 81
Konwersja danych na ciągi znaków oraz ciągów znaków na dane innych typów ... 83
Tworzenie tablic ... 84
Modyfikowanie tablic ... 85
Usuwanie elementów tablic ... 87
Przetwarzanie tablic w pętli ... 88
Funkcje obsługi tablic ... 90
Sortowanie tablic ... 91
Poruszanie się wśród elementów tablic ... 93
Zwijanie i rozwijanie tablic ... 95
Tworzenie zmiennych na podstawie indeksów tablic ... 96
Łączenie i rozdzielanie tablic ... 98
Porównywanie tablic ... 99
Wykonywanie działań na elementach tablic ... 101
Tworzenie tablic wielowymiarowych ... 103
Przetwarzanie tablic wielowymiarowych w pętli ... 105
Operatory tablicowe ... 106
Podsumowanie ... 108

Rozdział 4. Funkcje ... 111
Tworzenie funkcji ... 112
Przekazywanie danych do funkcji ... 113
Przekazywanie tablic do funkcji ... 115
Domyślne wartości argumentów ... 116
Przekazywanie argumentów przez adres ... 118
Tworzenie list argumentów o zmiennej długości ... 120
Wartości zwracane przez funkcje ... 122
Zwracanie tablic przez funkcje ... 123
Zwracanie list przez funkcje ... 125
Zwracanie adresów przez funkcje ... 127
Zasięg zmiennych ... 128
Zasięg globalny ... 130
Zmienne statyczne ... 132
Wykorzystanie funkcji, których nazwy zapisano w zmiennych ... 133
Tworzenie funkcji warunkowych ... 136
Tworzenie funkcji wewnątrz funkcji ... 137
Pliki włączane ... 138
Obsługa błędów zwracanych przez funkcje ... 140
Podsumowanie ... 141

Rozdział 5. Obsługa elementów sterujących HTML na stronach WWW 143
Przetwarzanie danych użytkownika w formularzach WWW 143
Tworzenie pól tekstowych 145
Pobieranie informacji z pól tekstowych 147
Tworzenie obszarów tekstowych 149
Tworzenie pól wyboru 151
Tworzenie przełączników 152
Tworzenie list wyboru 154
Tworzenie ukrytych elementów sterujących 156
Tworzenie pól do wprowadzania haseł 158
Tworzenie map obrazkowych 160
Wgrywanie plików na serwer 161
Odczytywanie wgranych plików 163
Tworzenie przycisków: sposób 1 165
Tworzenie przycisków: sposób 2 167
Tworzenie przycisków: sposób 3 169
Podsumowanie 171

Rozdział 6. Tworzenie formularzy na stronach WWW i sprawdzanie poprawności danych wprowadzanych przez użytkowników 173
Wyświetlanie wszystkich danych formularza za jednym razem 174
Przydatne zmienne serwera 176
Przydatne nagłówki HTTP 177
Sprawdzanie typu przeglądarki za pomocą nagłówków HTTP 178
Przekierowywanie użytkowników za pomocą nagłówków HTTP 180
Przekazywanie danych wprowadzanych w formularzach za pomocą tablic 182
Aplikacje WWW w jednym skrypcie PHP 183
Sprawdzanie poprawności danych wprowadzanych przez użytkowników 186
Sprawdzanie poprawności danych: pola obowiązkowe 187
Sprawdzanie poprawności danych: dane numeryczne 190
Sprawdzanie poprawności danych: ciągi znaków 191
Usuwanie znaczników HTML 193
Kodowanie znaczników HTML 194
Utrzymanie danych wprowadzonych wcześniej 197
Wykorzystanie JavaScript do sprawdzania poprawności danych 198
Zastosowanie uwierzytelniania HTTP 201
Podsumowanie 201

Rozdział 7. Programowanie obiektowe i operacje na plikach 203
Klasy i obiekty 203
Tworzenie klas 205
Tworzenie obiektów 207
Ograniczenia dostępu do właściwości i metod 208
Inicjowanie obiektów za pomocą konstruktorów 210
Tworzenie klas na podstawie innych klas: dziedziczenie 212
Zastosowanie chronionego dziedziczenia 214
Przesłanianie metod 216
Dostęp do metod klasy bazowej 217
Otwieranie plików: fopen 219
Czytanie wiersza tekstu z pliku: fgets 221
Czytanie pojedynczych znaków z pliku: fgetc 223
Czytanie danych z plików binarnych: fread 224
Czytanie całej zawartości pliku: file_get_contents 226
Analiza zawartości pliku: fscanf 227

Zapisywanie danych do pliku: fwrite ... 229
Dołączanie danych do pliku: fwrite ... 231
Zapisywanie całej zawartości pliku w jednej operacji: file_put_contents ... 233
Podsumowanie ... 234

Rozdział 8. Obsługa baz danych ... 237
Czym są bazy danych? ... 238
Podstawy języka SQL ... 238
Konfiguracja obsługi bazy danych w PHP ... 240
Tworzenie bazy danych za pomocą MySQL ... 241
Wprowadzanie danych do bazy danych ... 243
Dostęp do bazy danych MySQL z PHP ... 244
Wyświetlanie zawartości tabeli bazy danych ... 246
Aktualizowanie danych ... 248
Wprowadzanie nowych danych do bazy ... 250
Usuwanie danych ... 251
Tworzenie nowych tabel ... 253
Tworzenie bazy danych ... 255
Sortowanie danych ... 257
Pobranie modułu PEAR DB ... 259
Wyświetlanie zawartości tabeli za pomocą modułu DB ... 261
Wprowadzanie nowych danych do bazy za pomocą modułu DB ... 263
Aktualizacja danych za pomocą modułu DB ... 265
Podsumowanie ... 267

Rozdział 9. Pliki cookie, sesje, FTP i e-mail ... 269
Ustawianie plików cookie ... 270
Czytanie plików cookie ... 271
Ustawianie czasu ważności plików cookie ... 273
Usuwanie plików cookie ... 275
Obsługa protokołu FTP ... 276
FTP: pobieranie zawartości katalogu ... 277
FTP: pobieranie pliku ... 279
FTP: wgrywanie pliku na serwer ... 281
Wysyłanie wiadomości e-mail ... 283
Wysyłanie wiadomości e-mail z nagłówkami ... 285
Wysyłanie wiadomości e-mail z załącznikami ... 287
Obsługa sesji ... 288
Zapisywanie danych sesji ... 290
Tworzenie licznika odwiedzin ... 292
Wykorzystanie sesji bez plików cookie ... 294
Usuwanie danych z sesji ... 296
Podsumowanie ... 298

Dodatek A Elementy języka PHP ... 299

Dodatek B Funkcje PHP ... 317

Skorowidz ... 329

Rozdział 8.
Obsługa baz danych

W PHP jest wbudowana obsługa baz danych. To bardzo korzystne, ponieważ przechowywanie danych na serwerze stwarza programistom wielkie możliwości. Wiele typów serwerów baz danych jest obsługiwanych przez PHP. Listę zamieszczono w tabeli 8.1.

Tabela 8.1. *Obsługiwane bazy danych*

Baza danych		
Adabas	Ingres	Oracle
dBase	InterBase	Ovrimos
Empress	Front Base	PostgreSQL
FilePro	mSQL	Solid
Hyperwave Direct	MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	SQLite
Unix dbm		

Jak widać, obsługiwanych typów serwerów baz danych jest bardzo wiele. Nie możemy w tej książce opisać wszystkich, a zatem skoncentrujemy się na tym, który jest najbardziej popularny wśród programistów PHP — MySQL (bazę danych MySQL można pobrać za darmo ze strony *http://www.mysql.com*). Najnowsza, stabilna wersja bazy danych to MySQL 4.0. Właśnie z tej wersji skorzystamy w tym rozdziale.

W dalszej części rozdziału przyjrzymy się również *modułowi* PHP — `DB`, w którym zaimplementowano warstwę abstrakcji dla operacji z bazami danych. Dzięki temu można pracować z kilkoma różnymi serwerami baz danych (tabela 8.1), używając wywołań tych samych funkcji.

Podręczniki wbudowanej obsługi baz danych zestawionych w tabeli 8.1 można uzyskać pod adresem *http://www.php.net/nazwa_db*, gdzie `nazwa_db` oznacza nazwę bazy danych, na przykład *mysql*, *Sybase*, *mssql* itp. W przypadku baz danych ODBC należy zastosować nazwę *uodbc*; dla bazy danych Oracle — *oci8*. Moduł `DB` umożliwia dostęp do wszystkich typów baz danych w taki sam sposób, ale w przypadku skorzystania z wbudowanej obsługi, aplikacje działają znacznie szybciej.

Czym są bazy danych?

A zatem czym są bazy danych? Ogólna definicja jest bardzo prosta: bazy danych są mechanizmem umożliwiającym dostęp i przetwarzanie danych pod kontrolą systemu bazy danych lub aplikacji.

Najpopularniejszą konstrukcją w bazie danych jest *tabela*, od której zaczniemy omawianie pojęć związanych z bazami danych. Dla zobrazowania działania tabeli bazy danych przeanalizujmy przykład: wyobraźmy sobie nauczyciela, którego obowiązkiem jest zapisywanie ocen uczniów. Mógłby do tego wykorzystać tabelkę na kartce papieru podobną do tabeli 8.2.

Tabela 8.2. *Oceny uczniów*

Imię	Ocena
Anna	3
Marek	4
Edward	5
Franciszek	5
Tadeusz	5
Mateusz	4
Rafał	4
Tomasz	4

Powyższa tabelka odzwierciedla typową tabelę w bazie danych. Tabela komputerowa ma jednak przewagę nad papierową pod wieloma względami. Można ją sortować, indeksować, aktualizować i z łatwością organizować duże porcje danych (bez marnotrawienia dużych ilości papieru). Tabele można również łączyć na kilka różnych sposobów. Z połączenia tabel powstają tzw. *relacyjne* bazy danych.

Każda pojedyncza porcja informacji w tabeli, taka jak imię studenta, to pole. Zbiór pól, na przykład imię i ocena, tworzą rekord.

Każdy rekord odpowiada osobnemu wierszowi w tabeli, a każda kolumna reprezentuje oddzielne pole. Zbiór rekordów wierszy tworzy tabelę.

A zatem, czym jest baza danych? W najbardziej konwencjonalnej formie baza danych jest zbiorem tabel. Dostęp do danych w tych tabelach uzyskuje się za pomocą języka SQL. Zajmiemy się nim w następnym punkcie.

Podstawy języka SQL

Do komunikowania się z bazami danych w PHP wykorzystujemy język SQL (ang. *Structured Query Language*). Celem tego rozdziału jest zaprezentowanie wiadomości niezbędnych do posługiwania się językiem SQL w PHP. Załóżmy, że mamy tabelę

o nazwie owoc. Aby uzyskać jej kopię, dostępną do przetwarzania w kodzie PHP, należy wykonać następującą instrukcję SQL (tzw. *zapytanie*):

```
SELECT * FROM owoc
```

Do wykonania powyższego zapytania z poziomu PHP wykorzystamy funkcję mysql_query:

```
$query = "SELECT * FROM owoc";
$result = mysql_query($query)or die("Wykonanie zapytania nie powiodło się:
".mysql_error());
```

Dla wprawy przeanalizujemy kilka przykładów instrukcji SQL. Czytelnicy dobrze znający język SQL mogą pominąć ten fragment. Ci, którzy go nie znają, powinni uważnie przestudiować opisane tu zagadnienia, ponieważ ich znajomość będzie potrzebna w dalszej części tego rozdziału. Przyjmijmy, że w bazie danych jest tabela owoc, która ma dwa pola — nazwa (np. *jabłka* lub *pomarańcze*) oraz ilość (reprezentująca ilość owoców określonego rodzaju w magazynie).

Do pobrania pól z tabeli wykorzystamy instrukcję SELECT. Oto przykład instrukcji, w której symbol wieloznaczny * posłużył do pobrania wszystkich rekordów z tabeli owoc:

```
SELECT * FROM owoc
```

Instrukcja zwraca zestaw rekordów zawierający wszystkie wiersze tabeli owoc. Z tabeli można również pobierać określone pola. W naszym kolejnym przykładzie z tabeli owoc wybierzemy pola nazwa i ilość:

```
SELECT nazwa, ilość FROM owoc
```

Za pomocą klauzuli WHERE konfiguruje się kryteria wybierania rekordów, które muszą spełniać wiersze w zestawie rekordów generowanym przez zapytanie. Aby wybrać wszystkie rekordy z tabeli owoc, gdzie pole nazwa ma wartość *jabłka*, można skorzystać z następującej instrukcji:

```
SELECT * FROM owoc WHERE nazwa= "jabłka"
```

Oprócz znaku równości stosuje się dla pól także inne operatory porównań:

- < (mniejszy niż);
- <= (mniejszy bądź równy);
- > (większy niż);
- >= (większy bądź równy);

Za pomocą klauzuli IN określa się zbiór wartości pola spełniających warunki zapytania. Na przykład, aby wybrać rekordy, w których pole nazwa ma wartość *jabłka* lub *pomarańcze*, można wykorzystać instrukcję:

```
SELECT * FROM owoc WHERE nazwa IN ("jabłka", "pomarańcze")
```

W klauzulach instrukcji SQL stosuje się operacje logiczne. Oto przykład zapytania, w którym określono dwa kryteria: pole nazwa powinno zawierać wartość *jabłka* lub *pomarańcze*, a w polu ilość musi być jakaś liczba. Do sprawdzenia, czy pole nie jest puste, wykorzystamy słowo kluczowe NULL:

```
SELECT * FROM owoc WHERE nazwa IN ("jabłka", "pomarańcze") AND ilość IS NOT NULL
```

Do łączenia klauzul można wykorzystać operatory logiczne `AND`, `OR` oraz `NOT`. Zastosowanie operatora `AND` oznacza, że obie klauzule muszą być prawdziwe, `OR` — że dowolna z nich może być prawdziwa, natomiast operator `NOT` odwraca wartość klauzuli z prawdy na fałsz i z fałszu na prawdę.

Jak łatwo się domyślić, zestaw rekordów zwracany przez instrukcję SQL, może być uporządkowany. Oto przykład uporządkowania rekordów z tabeli `owoc` według pola `nazwa`:

```
SELECT * FROM owoc ORDER BY nazwa
```

Rekordy porządkuje się także malejąco. W tym celu stosuje się słowo kluczowe `DESC`:

```
SELECT * FROM owoc ORDER BY nazwa DESC
```

Rekordy można usunąć z bazy danych za pomocą instrukcji `DELETE`. W instrukcji pokazanej poniżej usuwamy wszystkie rekordy z tabeli `owoc`, których nazwy są różne od *jabłka* lub *pomarańcze*:

```
DELETE FROM owoc WHERE name NOT IN ("jabłka", "pomarańcze")
```

Aby wprowadzić zmiany w bazie danych, trzeba skorzystać z instrukcji `UPDATE`. Na przykład, aby zmienić wartość w polu `ilość` dla rekordu zawierającego ilość jabłek, należy skorzystać z następującej instrukcji:

```
UPDATE owoc SET ilość = "2006" WHERE nazwa = "jabłka"
```

Do tabeli można także wprowadzać nowe dane — oto przykład wprowadzenia nowego wiersza do tabeli `owoc`:

```
INSERT INTO owoc (nazwa, ilość) VALUES('morele', '203')
```

Teraz mamy pod ręką tyle narzędzi SQL, ile potrzeba. W następnym punkcie nauczymy się konfigurować obsługę serwera bazy danych w PHP.

Konfiguracja obsługi bazy danych w PHP

Konfiguracja obsługi bazy danych w PHP jest dość skomplikowana, a sposób wykonania tej czynności zależy od wykorzystywanego systemu operacyjnego. W tym punkcie opiszemy najbardziej typowe przypadki.

W systemach Unix, Linux i MacOS obsługę serwera bazy danych definiuje się podczas instalacji za pomocą opcji konfiguracyjnych zgodnie ze wskazówkami instalacji, które zawiera podręcznik online znajdujący się pod adresem *www.php.net/nazwabd*. Na przykład, zastosowanie opcji konfiguracji `--with-mysql [=KATALOG]` włącza obsługę bazy danych MySQL 4.0. Przy czym *KATALOG* oznacza katalog, w którym zainstalowano bazę danych MySQL. Domyślnie jest to */usr/local/mysql*. W przypadku bazy danych MySQL 4.1 (w czasie powstawania tej książki wersja 4.1 była testowana) należy skorzystać z takiej samej opcji, jak dla wersji 4.0 — `--with-mysql [=KATALOG]`.

Wówczas argument *KATALOG* nie ma wartości domyślnej i należy go ustawić na folder, w którym zapisano plik *mysql_config*. W przypadku bazy danych Oracle należy zastosować opcję `--with-oci8[=KATALOG]`, gdzie wartością domyślną argumentu *KATALOG* jest zawartość zmiennej środowiska `ORACLE_HOME`. Instalacje PHP u dostawców usług internetowych są zazwyczaj skonfigurowane do wykorzystania bazy danych MySQL, a także innych baz danych. Szczegółowych informacji można się dowiedzieć od pracowników pomocy technicznej.

W systemie Windows każdemu obsługiwanemu serwerowi baz danych odpowiada łączona dynamicznie biblioteka (plik DLL) zapisana w katalogu */ext*. Może to być na przykład katalog *c:\php\ext*. W tym katalogu należy odszukać plik DLL właściwy dla serwera bazy danych, którego obsługę chcemy włączyć i skopiować do głównego katalogu instalacji (na przykład *c:\php*). W przypadku bazy MySQL w wersji 4.0 lub wcześniejszej, jest to plik *php_mysql.dll*, w MySQL 4.1 i w nowszych plik nazywa się *php_mysqli.dll*, dla bazy danych Oracle — *php_oci8.dll*; dla Microsoft SQL Server — *mssql.dll*.

Po skopiowaniu pliku DLL należy uaktywnić obsługę serwera bazy danych poprzez usunięcie w pliku *php.ini* symbolu komentarza — średnika — z wiersza `extension=` odpowiadającego wybranej bazie danych. Oto sposób uaktywnienia obsługi bazy danych MySQL 4.0:

```
;extension=php_mime_magic.dll
;extension=php_mssql.dll
;extension=php_msql.dll
extension=php_mysql.dll
;ext.ension=php_oci8.dll
;extension=php_openssl.dll
;extension=php_oracle.dll
;extension=php_pdf.dll
;extension=php_pgsql.dll
        .
        .
        .
```

Po wprowadzeniu powyższych zmian konfiguracyjnych w systemie Windows należy zatrzymać serwer WWW i uruchomić go ponownie.

Tworzenie bazy danych za pomocą MySQL

W tym punkcie utworzymy przykładową bazę danych MySQL, korzystając z interfejsu wiersza polecenia. Aby uruchomić serwer bazy danych MySQL z wiersza polecenia (okno DOS w systemie Windows), należy przejść do katalogu *mysql/bin* i wprowadzić następujące polecenie:

```
%mysqld --console
```

Wykonanie polecenia spowoduje uruchomienie serwera. Teraz trzeba uruchomić sesję MySQL, w której połączymy się z serwerem. Otwieramy nowe okno wiersza polecenia i przechodzimy do katalogu *mysql/bin*. Jeśli wcześniej ustawiliśmy nazwę użytkownika i hasło, serwer MySQL uruchamiamy z wykorzystaniem opcji `-u` i `-p`:

```
%mysql -u user -p
Enter password: ********
```

Jeśli nie ustawiono nazwy użytkownika i hasła, wystarczy wprowadzić polecenie `mysql -u root` lub po prostu `mysql`:

```
%mysql -u root
Welcome to the MySQL monitor. Comnands end with ; or \g.
mysql>
```

Za symbolem zachęty `mysql>` wprowadzimy instrukcję `SELECT VERSION(), CURRENT_DATE;`. W ten sposób sprawdzimy, czy serwer MySQL działa:

```
mysql> SELECT VERSION(), CURRENT_DATE;
+-----------+--------------+
| VERSION() | CURRENT_DATE |
+-----------+--------------+
| 4.0.20a   | 2005-07-12   |
+-----------+--------------+
1 row in set (0.05 sec)
```

Aby zobaczyć, czy zdefiniowano jakieś bazy danych, można skorzystać z instrukcji `SHOW DATABASES`. Wyświetli się następujący wynik:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.01 sec)
```

W bazie danych MySQL są dwie wbudowane bazy danych — *mysql*, zawierająca dane administracyjne serwera MySQL, oraz *test* — przykładowa baza danych. W celu zapisania informacji dotyczących owoców i warzyw, utworzymy bazę danych pod nazwą `plody_rolne`. Wykorzystamy do tego instrukcję `CREATE DATABASE`:

```
mysql> CREATE DATABASE plody_rolne;
Query OK, 1 row affected (0.01 sec)
```

Aby utworzona baza danych stała się domyślna, należy wprowadzić polecenie `USE plody_rolne`:

```
mysql> USE plody_rolne;
Database changed
```

Jak sprawdzić, czy w bazie danych `plody_rolne` są jakieś tabele? Wystarczy użyć polecenia `SHOW TABLES`:

```
mysql> SHOW TABLES;
Empty set (0.01 sec)
```

W odpowiedzi uzyskaliśmy komunikat *Empty set*, co oznacza, że w bazie danych nie ma jeszcze tabel. Aby utworzyć tabelę owoc, należy zdefiniować jej pola danych. Pola mogą być różnego typu. Oto kilka z nich (w poniższym przykładzie wykorzystamy ciągi znaków — pola typu VARCHAR):

- VARCHAR(długość) — ciąg znaków o zmiennej długości;
- INT — liczba całkowita;
- DECIMAL(*całkowita_liczba_cyfr, liczba_miejsc_dziesiętnych*) — liczba zmiennoprzecinkowa;
- DATETIME — obiekty typu data i godzina, na przykład 2006-11-15 20:00:00.

Poniżej zamieszczono instrukcję tworzącą tabelę owoc z polami nazwa i ilość typu ciąg znaków:

```
mysql> CREATE TABLE owoc (nazwa VARCHAR(20), ilosc VARCHAR(20));
Query OK, 0 rows affected (0.13 sec)
mysql> SHOW TABLES;
+-----------------------+
| Tables_in_plody_rolne |
+-----------------------+
| owoc                  |
+-----------------------+
1 row in set (0.00 sec)
```

Aby wyświetlić opis nowej tabeli, można skorzystać z polecenia DESCRIBE:

```
mysql> DESCRIBE owoc;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| nazwa | varchar(20) | YES  |     | NULL    |       |
| ilosc | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.03 sec)
```

Wprowadzanie danych do bazy danych

W poprzednim punkcie utworzyliśmy bazę danych MySQL plody_rolne i zdefiniowaliśmy tabelę owoc do zapisywania danych na temat różnych owoców. W tym punkcie do tabeli bazy danych owoc wprowadzimy dane zestawione w tabeli 8.3.

Tabela 8.3. *Dane tabeli owoc*

Nazwa	Ilość
jabłka	1020
pomarańcze	3329
banany	8582
gruszki	235

W celu wprowadzenia tych danych do tabeli owoc będziemy kontynuować sesję MySQL rozpoczętą w poprzednim punkcie lub rozpoczniemy nową w oknie wiersza polecenia:

```
$mysql -u root
Welcome to the MySQL monitor. Commands end with; or \g.
```

Przejdziemy do bazy danych plody_rolne i za pomocą instrukcji INSERT załadujemy dane do tabeli:

```
mysql> USE plody_rolne
Database changed
mysql> INSERT INTO owoc VALUES ('jabłka', '1020');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO owoc VALUES ('pomarańcze', '3329');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO owoc VALUES ('banany', '8582');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO owoc VALUES ('gruszki', '235');
Query OK, 1 row affected (0.00 sec)
```

To wszystko. Dla sprawdzenia, czy dane zostały wprowadzone poprawnie, można wyświetlić zawartość tabeli owoc za pomocą instrukcji SELECT. Sesję kończy się za pomocą polecenia quit:

```
mysql> SELECT * FROM owoc;
+------------+-------+
| nazwa      | ilosc |
+------------+-------+
| jabłka     | 1020  |
| pomarańcze | 3329  |
| banany     | 8582  |
| gruszki    | 235   |
+------------+-------+
4 rows in set (0.02 sec)
mysql>quit
```

Dostęp do bazy danych MySQL z PHP

Nadszedł czas, aby skorzystać z utworzonej bazy danych z poziomu PHP. W tym celu należy uruchomić serwer MySQL (chyba że korzystamy z usług dostawcy Internetu, gdzie serwer MySQL działa przez cały czas). Jak już wspominaliśmy, w tym celu należy uruchomić demona *mysqld* w katalogu *mysql/bin*:

```
%mysqld --console
```

Aby zatrzymać serwer, w katalogu *mysql/bin*, w innym oknie wiersza polecenia, należy wpisać:

```
%mysqladmin -u root shutdown
```

Do połączenia się z bazą danych MySQL służy funkcja `mysql_connect`. Jej argumentami są nazwa hosta oraz opcjonalnie nazwa użytkownika i hasło:

```
$connection = mysql_connect("localhost","root","")
    or die ("Nie można połączyć się z serwerem");
```

Po połączeniu można przekazać wartość zmiennej `$connection` do funkcji `mysql_select_db` w celu wybrania roboczej bazy danych. W naszym przypadku jest to baza danych `plody_rolne`:

```
$connection = mysql_connect("localhost","root","")
    or die ("Nie można połączyć się z serwerem");
$db = mysql_select_db("plody_rolne", $connection)
    or die ("Nie można wybrać bazy danych");
```

Właśnie połączyliśmy się z bazą danych `plody_rolne`. Aby pobrać wszystkie dane z tabeli `owoc`, można wykonać poniższe zapytanie SQL za pomocą funkcji `mysql_query` — warto także zauważyć, że funkcja `mysql_error` zwraca komunikat o błędzie bazy danych MySQL:

```
$query = "SELECT * FROM owoc";
$result = mysql_query($query)
    or die("Wykonanie zapytania nie powiodło się: ". mysql_error());
```

Oprócz funkcji `mysql_connect`, `mysql_select_db` i `mysql_query`, dostępne są inne funkcje obsługujące bazę danych MySQL. Kilka z nich wymieniono poniżej (listę funkcji wspomagających obsługę innych serwerów baz danych można uzyskać pod adresem *www.php.net/**nazwa_bd***, gdzie ***nazwa_bd*** jest nazwą określonego serwera):

- `mysql_affected_rows` — pobiera liczbę wierszy, których dotyczyła poprzednia operacja SQL;
- `mysql_change_user` — zmiana użytkownika;
- `mysql_client_encoding` — zwraca nazwę bieżącego zestawu znaków;
- `mysql_close` — zamyka połączenia MySQL;
- `mysql_connect` — otwiera połączenie z serwerem MySQL;
- `mysql_create_db` — tworzy bazę danych MySQL;
- `mysql_data_seek` — wyszukuje dane w bazie danych;
- `mysql_db_name` — pobiera nazwę bazy danych;
- `mysql_db_query` — przesyła zapytanie SQL;
- `mysql_drop_db` — usuwa bazę danych MySQL;
- `mysql_error` — zwraca tekst komunikatu o błędzie ostatnio wykonywanej operacji MySQL;
- `mysql_fetch_array` — zwraca wiersz wyniku w postaci tablicy asocjacyjnej, tablicy z indeksami liczbowymi lub obu;
- `mysql_fetch_assoc` — pobiera wiersz wyniku w postaci tablicy asocjacyjnej;

- `mysql_fetch_row` — pobiera wiersz wyniku w postaci tablicy z indeksami liczbowymi;
- `mysql_field_len` — zwraca długość określonego pola;
- `mysql_field_name` — zwraca nazwę określonego pola w zestawie wyników;
- `mysql_field_seek` — przemieszcza wskaźnik pola do określonego przesunięcia;
- `mysql_field_table` — zwraca nazwę tabeli, w której jest określone pole;
- `mysql_field_type` — zwraca typ wybranego pola w zestawie wyników;
- `mysql_get_server_info` — pobiera informacje o serwerze MySQL;
- `mysql_info` — pobiera informacje o ostatnim zapytaniu;
- `mysql_list_dbs` — wyświetla bazy danych dostępne na serwerze MySQL;
- `mysql_list_fields` — wyświetla pola w tabeli bazy danych MySQL;
- `mysql_list_tables` — wyświetla listę tabel w bazie danych MySQL;
- `mysql_num_fields` — pobiera liczbę pól zwróconych w wyniku zapytania;
- `mysql_num_rows` — pobiera liczbę wierszy zapytania;
- `mysql_pconnect` — otwiera trwałe połączenie z serwerem MySQL;
- `mysql_query` — wysyła zapytanie SQL;
- `mysql_result` — pobiera dane wyniku;
- `mysql_select_db` — wybiera bazę danych MySQL;
- `mysql_tablename` — pobiera nazwę tabeli określonego pola.

Wyświetlanie zawartości tabeli bazy danych

Jesteśmy gotowi, aby zaprezentować wspólne działanie SQL i PHP w przeglądarce. Najpierw spróbujemy pobrać tabelę `owoc` z bazy danych `plody_rolne` i wyświetlić je jako tabelę HTML. Przede wszystkim trzeba połączyć się z serwerem bazy danych, wybrać bazę i utworzyć obiekt `$result` reprezentujący tabelę `owoc`:

```
$connection = mysql_connect("localhost","root","")
    or die ("Nie można połączyć się z serwerem");
$db = mysql_select_db("plody_rolne", $connection)
    or die ("Nie można wybrać bazy danych");

$query = "SELECT * FROM owoc";
$result = mysql_query($query)
    or die("Wykonanie zapytania nie powiodło się: " . mysql_error());
```

Teraz można pobrać kolejne wiersze z tabeli w pętli `while` za pomocą funkcji `mysql_fetch_array` i odczytać pola `nazwa` i `ilosc` każdego wiersza w następujący sposób:

```
while ($row = mysql_fetch_array($result))
{
    echo "<TR>";
    echo "<TD>", $row['nazwa'], "</TD><TD>", $row['ilosc'], "</TD>";
    echo "</TR>";
}
```

Pozostało jeszcze napisanie kodu HTML tabeli, w której wyświetlimy dane. Kompletny kod przykładu — skrypt *phpdatatable.php* — zamieszczono na listingu 8.1. Zwróćmy uwagę, że na koniec zamknęliśmy połączenie za pomocą funkcji `mysql_close`. Zawsze należy o tym pamiętać po zakończeniu wykonywania operacji z bazą danych.

Listing 8.1. *Wyświetlanie tabeli bazy danych, phpdatatable.php*

```
<HTML>
    <HEAD>
        <TITLE>
            Wyświetlanie tabel bazy danych MySQL
        </TITLE>
    </HEAD>
    <BODY>
        <CENTER>
            <H1>Wyświetlanie tabel bazy danych MySQL</H1>

            <?php
                $connection = mysql_connect("localhost","root","")
                    or die ("Nie można połączyć się z serwerem");
                $db = mysql_select_db("plody_rolne", $connection)
                    or die ("Nie można wybrać bazy danych");

                $query = "SELECT * FROM owoc";
                $result = mysql_query($query)
                    or die("Wykonanie zapytania nie powiodło się: ".mysql_error());

                echo "<TABLE BORDER='1'>";
                echo "<TR>";
                echo "<TH>Nazwa</TH><TH>Ilość</TH>";
                echo "</TR>";

                while ($row = mysql_fetch_array($result))
                {
                    echo "<TR>";
                    echo "<TD>", $row['nazwa'], "</TD><TD>", $row['ilosc'],
                        "</TD>";
                    echo "</TR>";
                }
                echo "</TABLE>";

                mysql_close($connection);
            ?>
        </CENTER>
    </BODY>
</HTML>
```

Wynik działania skryptu — zawartość tabeli `owoc` — pokazano na rysunku 8.1. Super!



Rysunek 8.1. *Wyświetlanie zawartości tabeli bazy danych*

To tyle. Udało się nam utworzyć bazę danych MySQL i pobrać z niej dane, a następnie wyświetlić je na stronie WWW.

Aktualizowanie danych

W przypadku, gdy dane zmienią się, można z łatwością zaktualizować bazę danych za pomocą odpowiedniej instrukcji SQL. Oto przykład: załóżmy, że ktoś kupił kilogram gruszek, w związku z czym ich ilość w magazynie zmniejszyła się z 235 na 234 kilogramy. Aby zaktualizować tę wartość, należy podłączyć się do bazy danych MySQL i wybrać bazę danych `plody_rolne`:

```
$connection = mysql_connect("localhost","root","")
    or die ("Nie można połączyć się z serwerem");

$db = mysql_select_db("plody_rolne",$connection)
    or die ("Nie można wybrać bazy danych");
```

Zobaczmy, jak można zaktualizować wartość pola `ilosc` w tabeli `owoc` z 235 na 234 w wierszu zawierającym dane dla gruszek:

```
$query = "UPDATE owoc SET ilosc = 234 WHERE nazwa = 'gruszki'";
        .
        .
        .
```

Wystarczy teraz wykonać to zapytanie:

```
$query = "UPDATE owoc SET ilosc = 234 WHERE nazwa = 'gruszki'";
$result = mysql_query($query)
    or die("Wykonanie zapytania nie powiodło się: ".mysql_error());
```

To wszystko. Skrypt *phpdataupdate.php* zamieszczono na listingu 8.2.

Listing 8.2. *Aktualizacja danych w bazie, phpdataupdate.php*

```
<HTML>
    <HEAD>
        <TITLE>
            Aktualizacja danych w bazie
        </TITLE>
    </HEAD>
    <BODY>
        <CENTER>
            <H1>Aktualizacja danych w bazie</H1>

            <?php
            $connection = mysql_connect("localhost","root","")
                or die ("Nie można połączyć się z serwerem");
            $db = mysql_select_db("plody_rolne",$connection)
                or die ("Nie można wybrać bazy danych");

            $query = "UPDATE owoc SET ilosc = 234 WHERE nazwa = 'gruszki'";
            $result = mysql_query($query)
                or die("Wykonanie zapytania nie powiodło się: ".mysql_error());

            $query = "SELECT * FROM owoc";
            $result = mysql_query($query)
                or die("Wykonanie zapytania nie powiodło się: " . mysql_error());

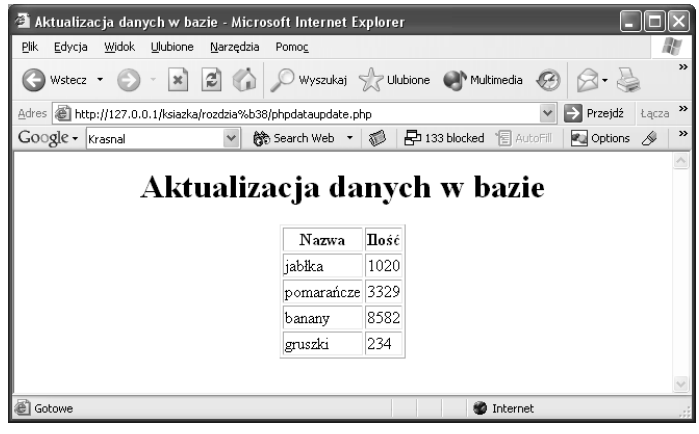
            echo "<TABLE BORDER='1'>";
            echo "<TR>";
            echo "<TH>Nazwa</TH><TH>Ilość</TH>";
            echo "</TR>";

            while ($row = mysql_fetch_array($result))
            {
                echo "<TR>";
                echo "<TD>", $row['nazwa'], "</TD><TD>",
                    $row['ilosc'], "</TD>";
                echo "</TR>";
            }
            echo "</TABLE>";

            mysql_close($connection);
        ?>
        </CENTER>
    </BODY>
</HTML>
```

Nową zawartość tabeli owoc zaprezentowano na rysunku 8.2. Jak można zauważyć, ilość gruszek zmniejszyła się z 235 do 234.

Rysunek 8.2.
Aktualizacja danych



Wprowadzanie nowych danych do bazy

Co zrobić, jeśli do magazynu przyjmiemy nowy towar, na przykład nowy rodzaj owoców? W jaki sposób wprowadzić nowy element do tabeli bazy danych? Pokażemy to w naszym kolejnym przykładzie, gdzie do tabeli `owoc` wprowadzimy nowy rodzaj owoców — morele. Najpierw połączymy się z bazą danych `plody_rolne`:

```
$connection = mysql_connect("localhost","root","")
    or die ("Nie można połączyć się z serwerem");

$db = mysql_select_db("plody_rolne",$connection)
    or die ("Nie można wybrać bazy danych");
```

Nowy wiersz dla moreli utworzymy za pomocą instrukcji SQL `INSERT`:

```
$query = "INSERT INTO owoc (nazwa, ilość) VALUES('morele', '203')";
        .
        .
        .
```

Aby wprowadzić nowy wiersz, wystarczy wykonać to zapytanie:

```
$query = "INSERT INTO owoc (nazwa, ilość) VALUES('morele', '203')";
$result = mysql_query($query)
    or die("Wykonanie zapytania nie powiodło się: ". mysql_error());
```

Po wprowadzeniu nowego wiersza można przejrzeć nową zawartość tabeli tak, jak pokazano w skrypcie *phpdatainsert.php*, zamieszczonym na listingu 8.3.

Listing 8.3. *Wprowadzanie nowych danych do bazy, phpdatainsert.php*

```
<HTML>
    <HEAD>
        <TITLE>
            Wprowadzanie nowych danych do bazy
        </TITLE>
    </HEAD>
```

```
    <BODY>
        <CENTER>
            <H1>Wprowadzanie nowych danych</H1>

            <?php
                $connection = mysql_connect("localhost","root","")
                    or die ("Nie można połączyć się z serwerem");
                $db = mysql_select_db("plody_rolne", $connection)
                    or die ("Nie można wybrać bazy danych");

                $query = "INSERT INTO owoc (nazwa, ilosc) VALUES('morele', '203')";
                $result = mysql_query($query)
                    or die("Wykonanie zapytania nie powiodło się: ".
                    mysql_error());

                $query = "SELECT * FROM owoc";
                $result = mysql_query($query)
                    or die("Wykonanie zapytania nie powiodło się: ".
                    mysql_error());

                echo "<TABLE BORDER='1'>";
                echo "<TR>";
                echo "<TH>Nazwa</TH><TH>Ilość</TH>";
                echo "</TR>";

                while ($row = mysql_fetch_array($result))
                {
                    echo "<TR>";
                    echo "<TD>", $row['nazwa'], "</TD><TD>",
                                $row['ilosc'], "</TD>";
                    echo "</TR>";
                }
                echo "</TABLE>";

                mysql_close($connection);
            ?>
        </CENTER>
    </BODY>
</HTML>
```

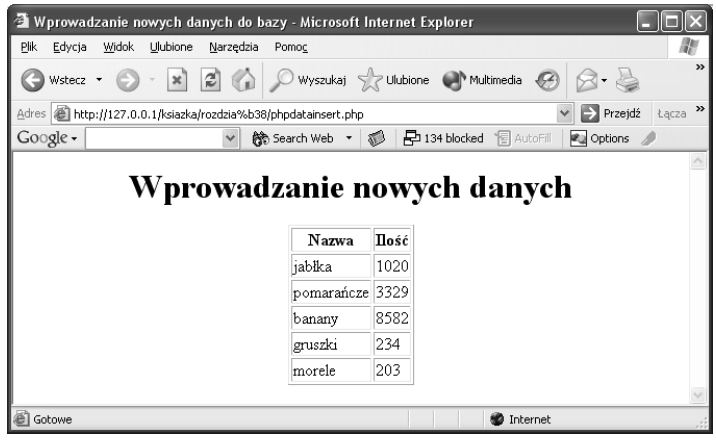
Rysunek 8.3 przedstawia rezultat działania skryptu. Jak widać, bez trudu udało się wprowadzić morele do bazy danych.

Usuwanie danych

W jaki sposób usuwa się dane? Przyjmijmy, że dostawca moreli nie dostarcza towaru, a zatem trzeba usunąć zapis w bazie danych utworzony w poprzednim punkcie. Jak zwykle, zaczniemy od połączenia z bazą danych:

```
$connection = mysql_connect("localhost","root","")
    or die ("Nie można połączyć się z serwerem");

$db = mysql_select_db("plody_rolne",$connection)
    or die ("Nie można wybrać bazy danych");
```



Rysunek 8.3. *Wprowadzanie danych do bazy*

Następnie zastosujemy instrukcję SQL `DELETE` i usuniemy wiersz dotyczący moreli. Wiersz do usunięcia zidentyfikujemy na podstawie nazwy. Oto potrzebna instrukcja SQL:

```
$query = "DELETE FROM owoc WHERE nazwa = 'morele'";
        .
        .
        .
```

Aby usunąć wiersz dotyczący moreli, wystarczy wykonać powyższe zapytanie:

```
$query = "DELETE FROM owoc WHERE nazwa = 'morele'";
$result = mysql_query($query)
    or die("Wykonanie zapytania nie powiodło się: " . mysql_error());
```

Po usunięciu wiersza dotyczącego moreli wyświetlimy nową zawartość tabeli `owoc`, tak jak pokazano w skrypcie *phpdatadelete.php*, zamieszczonym na listingu 8.4.

Listing 8.4. *Usuwanie danych, phpdatadelete.php*

```
<HTML>
    <HEAD>
        <TITLE>
            Usuwanie danych z bazy danych MySQL </TITLE>
    </HEAD>
    <BODY>
        <CENTER>
            <H1>Usuwanie danych z bazy danych MySQL</H1>

            <?php
                $connection = mysql_connect("localhost","root","")
                    or die ("Nie można połączyć się z serwerem");
                $db = mysql_select_db("plody_rolne",$connection)
                    or die ("Nie można wybrać bazy danych");

                $query = "DELETE FROM owoc WHERE nazwa = 'morele'";
                $result = mysql_query($query)
                    or die("Wykonanie zapytania nie powiodło się: ". mysql_error());

                $query = "SELECT * FROM owoc";
```

```
        $result = mysql_query($query)
            or die("Wykonanie zapytania nie powiodło się: " . mysql_error());

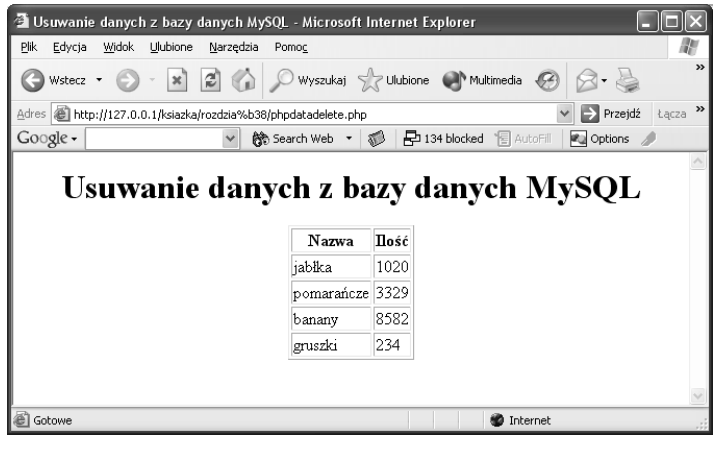
        echo "<TABLE BORDER='1'>";
        echo "<TR>";
        echo "<TH>Nazwa</TH><TH>Ilość</TH>";
        echo "</TR>";

        while ($row = mysql_fetch_array($result))
        {
            echo "<TR>";
            echo "<TD>", $row['nazwa'], "</TD><TD>",
                         $row['ilosc'], "</TD>";
            echo "</TR>";
        }
        echo "</TABLE>";

        mysql_close($connection);
    ?>
  </CENTER>
 </BODY>
</HTML>
```

Wyniki działania skryptu pokazano na rysunku 8.4. Rzeczywiście, dane dotyczące moreli nie są widoczne. W ten sposób nauczyliśmy się wprowadzać dane do tabel bazy danych i je z nich usuwać.

Rysunek 8.4.
Usuwanie danych



Tworzenie nowych tabel

Co zrobić, aby utworzyć w bazie danych nową tabelę? Załóżmy, że interesy idą dobrze i oprócz owoców, chcemy zacząć sprzedawać warzywa. Czy można „w locie" utworzyć tabelę `warzywa`? Nie ma problemu. Do utworzenia nowej tabeli służy instrukcja SQL `CREATE TABLE`. Zobaczmy, jak można utworzyć nową tabelę `warzywa` zawierającą pola `nazwa` i `ilosc`:

```
$query = "CREATE TABLE warzywa (nazwa VARCHAR(20), ilosc VARCHAR(20))";
$result = mysql_query($query)
    or die("Wykonanie zapytania nie powiodło się: " . mysql_error());
```

Teraz, można wprowadzić dane do nowej tabeli `warzywa`, za pomocą instrukcji `INSERT`:

```
$query = "INSERT INTO warzywa (nazwa, ilosc) VALUES('kukurydza', '2083')";
$result = mysql_query($query)
    or die("Wykonanie zapytania nie powiodło się: " . mysql_error());
```

Kompletny kod zamieszczono w skrypcie *phpdatacreate.php*, na listingu 8.5.

Listing 8.5. *Tworzenie nowej tabeli, phpdatacreate.php*

```
<HTML>
    <HEAD>
        <TITLE>Tworzenie nowej tabeli</TITLE>
    </HEAD>
    <BODY>
        <CENTER>
            <H1>Tworzenie nowej tabeli</H1>
            <?php
                $connection = mysql_connect("localhost","root","")
                    or die ("Nie można połączyć się z serwerem");
                $db = mysql_select_db("plody_rolne",$connection)
                    or die ("Nie można wybrać bazy danych");

                $query = "CREATE TABLE warzywa (nazwa VARCHAR(20),
                    ilosc VARCHAR(20))";
                $result = mysql_query($query)
                    or die("Wykonanie zapytania nie powiodło się: ". mysql_error());

                $query = "INSERT INTO warzywa (nazwa, ilosc) VALUES(
                    'kukurydza', '2083')";
                $result = mysql_query($query)
                    or die("Wykonanie zapytania nie powiodło się: " . mysql_error());

                $query = "INSERT INTO warzywa (nazwa, ilosc)
                    VALUES('szpinak', '1993')";
                $result = mysql_query($query)
                    or die("Wykonanie zapytania nie powiodło się: ". mysql_error());

                $query = "INSERT INTO warzywa (nazwa, ilosc)
                    VALUES('buraki', '437')";
                $result = mysql_query($query)
                    or die("Wykonanie zapytania nie powiodło się: " . mysql_error());

                $query = "SELECT * FROM warzywa";
                $result = mysql_query($query)
                   or die("Wykonanie zapytania nie powiodło się: ". mysql_error());

                echo "<TABLE BORDER='1'>";
                echo "<TR>";
                echo "<TH>Nazwa</TH><TH>Ilość</TH>";
                echo "</TR>";

                while ($row = mysql_fetch_array($result))
```

```
                {
                    echo "<TR>";
                    echo "<TD>", $row[nazwa], "</TD><TD>".
                    $row['ilosc'], "</TD>";
                    echo "</TR>";
                }
                echo "</TABLE>";

                mysql_close($connection);
            ?>
        </CENTER>
    </BODY>
</HTML>
```

Zawartość nowej tabeli można zobaczyć na rysunku 8.5. Doskonale.



Rysunek 8.5. *Tworzenie nowej tabeli*

Tworzenie bazy danych

Z poziomu PHP można nawet utworzyć całą bazę danych. Oto przykład utworzenia bazy danych `zywnosc` za pomocą polecenia `CREATE DATABASE`:

```
$query = "CREATE DATABASE IF NOT EXISTS zywnosc";
$result = mysql_query($query)
    or die("Wykonanie zapytania nie powiodło się: ". mysql_error());
```

Do bazy danych dodamy nową tabelę — `przekaski`:

```
$db = mysql_select_db("zywnosc",$connection)
    or die ("Nie można wybrać bazy danych");

$query = "CREATE TABLE przekaski (nazwa VARCHAR(20), ilosc VARCHAR(20))";
$result = mysql_query($query)
    or die("Wykonanie zapytania nie powiodło się: ". mysql_error());
```

Pozostaje jeszcze wprowadzenie kilku wierszy za pomocą instrukcji INSERT na wzór skryptu *phpdatacreatedb.php*, zamieszczonego na listingu 8.6.

Listing 8.6. *Tworzenie nowej bazy danych, phpdatacreatedb.php*

```
<HTML>
    <HEAD>
        <TITLE>Tworzenie nowej bazy danych</TITLE>
    </HEAD>
    <BODY>
        <CENTER><H1>Tworzenie nowej bazy danych</H1>
            <?php
                $connection = mysql_connect("localhost","root","")
                    or die ("Nie można połączyć się z serwerem");

                $query = "CREATE DATABASE IF NOT EXISTS zywnosc";
                $result = mysql_query($query)
                    or die("Wykonanie zapytania nie powiodło się: ".
                    mysql_error());

                $db = mysql_select_db("zywnosc",$connection)
                    or die ("Nie można wybrać bazy danych");

                $query = "CREATE TABLE przekaski (nazwa VARCHAR(20), ilosc
                    VARCHAR(20))";
                $result = mysql_query($query)
                    or die("Wykonanie zapytania nie powiodło się: " . mysql_error());

                $query = "INSERT INTO przekaski (nazwa, ilosc)
                    VALUES('chipsy', '2843')";
                $result = mysql_query($query)
                   or die("Wykonanie zapytania nie powiodło się: " . mysql_error());

                $query = "INSERT INTO przekaski (nazwa, ilosc) VALUES('pizza',
                    '1955')";
                $result = mysql_query($query)
                   or die("Wykonanie zapytania nie powiodło się: ". mysql_error());
                $query = "INSERT INTO przekaski(nazwa, ilosc)
                    VALUES('cheeseburgery', '849')";
                $result = mysql_query($query)
                  or die("Wykonanie zapytania nie powiodło się: ". mysql_error());

                $query = "SELECT * FROM przekaski";
                $result = mysql_query($query)
                  or die("Wykonanie zapytania nie powiodło się: " . mysql_error());

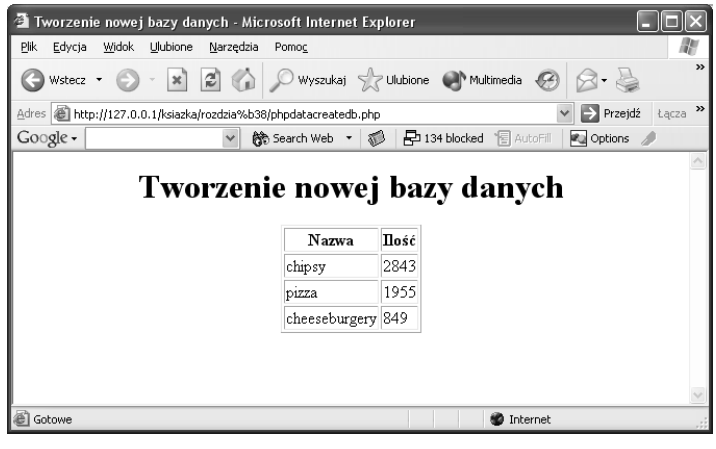
                echo "<TABLE BORDER='1'>";
                echo "<TR>";
                echo "<TH>Nazwa</TH><TH>Ilość</TH>";
                echo "</TR>";

                while ($row = mysql_fetch_array($result))
                {
                    echo "<TR>";
                    echo "<TD>", $row['nazwa'], "</TD><TD>",
                        $row['ilosc'], "</TD>";
```

```
                echo "</TR>";
            }
            echo "</TABLE>";
            mysql_close($connection);
        ?>
    </CENTER>
  </BODY>
</HTML>
```

Rezultat działania powyższego skryptu — zawartość nowej bazy danych — ilustruje rysunek 8.6.

Rysunek 8.6. *Tworzenie nowej bazy danych*



Sortowanie danych

Czy można posortować dane? Nic łatwiejszego. Wystarczy połączyć się z bazą danych tak, jak zwykle:

```
$connection = mysql_connect("localhost","root","")
    or die ("Nie można połączyć się z serwerem");

$db = mysql select db("plody_rolne",$connection)
    or die ("Nie można wybrać bazy danych");
```

Po połączeniu można użyć klauzuli `ORDER BY` i określić pole, według którego chcemy sortować. Na przykład, aby posortować owoce według nazwy, można zastosować taką instrukcję:

```
$query = "SELECT * FROM owoc ORDER BY nazwa";
        .
        .
        .
```

Zapytanie wykonujemy podobnie, jak wcześniej:

```
$query = "SELECT * FROM owoc ORDER BY nazwa";

$result = mysql_query($query)
    or die("Wykonanie zapytania nie powiodło się: ". mysql_error());
```

Efekt sortowania można obejrzeć, wykonując skrypt *phpdatasort.php*, zamieszczony na listingu 8.7.

Listing 8.7. *Sortowanie danych, phpdatasort.php*

```
<HTML>
    <HEAD>
        <TITLE>
            Sortowanie danych
        </TITLE>
    </HEAD>
    <BODY>
        <CENTER>
            <H1>
                Sortowanie danych
            </H1>

            <?php
                $connection = mysql_connect("localhost","root","")
                    or die ("Nie można połączyć się z serwerem");
                $db = mysql_select_db("plody_rolne",$connection)
                    or die ("Nie można wybrać bazy danych");

                $query = "SELECT * FROM owoc ORDER BY nazwa";
                $result = mysql_query($query)
                    or die("Wykonanie zapytania nie powiodło się: ".mysql_error());

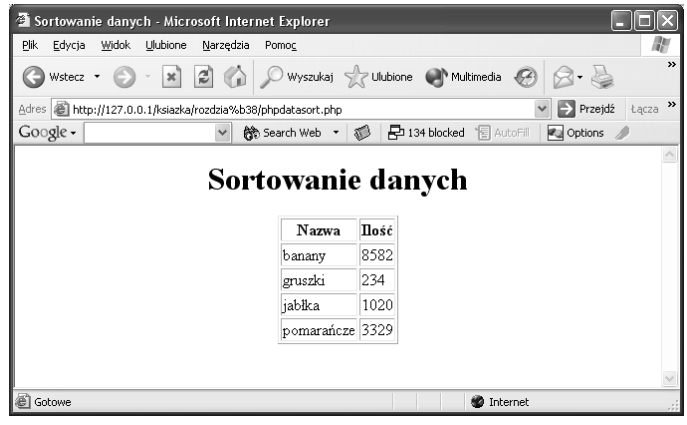
                echo "<TABLE BORDER='1'>";
                echo "<TR>";
                echo "<TH>Nazwa</TH><TH>Ilość</TH>";
                echo "</TR>";

                while ($row = mysql_fetch_array($result))
                {
                    echo "<TR>";
                    echo "<TD>", $row['nazwa'], "</TD><TD>",
                        $row['ilosc'], "</TD>";
                    echo "</TR>";
                }
                echo "</TABLE>";

                mysql_close($connection);
            ?>
        </CENTER>
    </BODY>
</HTML>
```

Wyniki działania skryptu zaprezentowano na rysunku 8.7. Jak łatwo zauważyć, dane z tabeli `owoc` zostały posortowane według nazwy. Sortowanie danych przydaje się do przygotowywania danych, przeznaczonych do wyświetlenia w przeglądarce użytkownika.

Rysunek 8.7.
Sortowanie danych



Pobranie modułu PEAR DB

Oprócz indywidualnej obsługi różnych serwerów baz danych, w PHP jest dostępny moduł *DB*, oferujący poziom abstrakcji i ukrywający detale pracy z poszczególnymi serwerami baz danych. Jeśli zdecydujemy się na skorzystanie z modułu DB, będziemy mogli stosować te same funkcje do pracy z różnymi serwerami baz danych. W przypadku zmiany serwera w kodzie będzie trzeba jedynie wprowadzić jego nazwę.

DB jest rozszerzeniem PHP należącym do repozytorium PEAR (*PHP Extension and Application Repository*). Jeśli kompilujemy PHP z kodów źródłowych, obsługa repozytorium PEAR jest zainstalowana jako skrypt *pear.php* w katalogu instalacji PHP. Dystrybucje binarne zawierają skrypt, który instaluje obsługę PEAR z witryny *go-pear.org*. Aby uruchomić rozszerzenie PEAR, należy przejść do katalogu z instalacją PHP i w wierszu polecania wpisać polecenie *pear*.

W systemie Windows obsługa PEAR jest dostępna w wersji instalowanej ręcznie (nie jest natomiast dostępna w wersji dostarczanej jako plik instalatora systemu Windows). W katalogu z dystrybucją PHP (np. *C:\PHP*) powinien być plik *go-pear.bat*. Aby go uruchomić, należy wpisać *go-pear* w wierszu polecenia lub dwukrotnie kliknąć skrypt *go-pear.bat*. Po uruchomieniu wyświetli się kilka pytań, a następnie utworzy się skrypt *pear.bat* służący do uruchamiania repozytorium PEAR.

Repozytorium PEAR składa się z wielu modułów. Jednym z nich jest moduł DB ładowany domyślnie w czasie ładowania repozytorium. Jeśli zatem repozytorium PEAR jest zainstalowane, moduł DB również powinien być zainstalowany (w systemie Windows po uruchomieniu skryptu *go-pear.bat* wyświetla się pytanie, czy chcemy zainstalować moduł DB). Aby sprawdzić, jakie moduły PEAR zostały zainstalowane, wystarczy uruchomić PEAR z opcją list:

```
%php>pear list
INSTALLED PACKAGES:
===================
PACKAGE          VERSION STATE
```

```
Archive_Tar     1.2     stable
Console_Getopt  1.2     stable
DB              1.6.5   stable
Mail            1.1.3   stable
Net_SMTP        1.2.6   stable
Net_Socket      1.0.2   stable
PEAR            1.3.1   stable
PHPUnit         1.0.0   stable
XML_Parser      1.2.0   stable
XML_RPC         1.1.0   stable
```

Jest to lista zainstalowanych modułów PEAR. Jak widać, wśród nich jest moduł DB. Jeśli na liście zainstalowanych modułów nie ma modułu DB, można go zainstalować w następujący sposób:

```
%pear install DB
```

Aby dowiedzieć się, jakie inne moduły są dostępne w repozytorium PEAR, należy skorzystać z polecenia `list-all`:

```
%pear list-all
ALL PACKAGES:
=============
PACKAGE                 LATEST   LOCAL
APC                     2.0.4
Cache                   1.5.4
Cache_Lite              1.3.1
apd                     1.0
memcache                1.3
perl                    0.6
PHPUnit                 1.0.1    1.0.1
PHPUnit2                2.0.2
PHP_Compat              1.1.0
Var_Dump                1.0.0
Xdebug                  1.3.2
Archive_Tar             1.2      1.2
bz2                     1.0
Contact_Vcard_Build     1.1
Contact_Vcard_Parse     1.30
File_Fstab              2.0.0
File_HtAccess           1.1.0
File_Passwd             1.1.0
File_SMBPasswd          1.0.0
MP3_ID                  1.1.3
zip                     1.0
Auth                    1.2.3
Auth_HTTP               2.0
Auth_PrefManager        1.1.3
Auth_RADIUS             1.0.4
Auth_SASL               1.0.1
radius                  1.2.4
Benchmark               1.2.1
Config                  1.10.2
    .
    .
    .
```

Zawartość pliku pomocy dla repozytorium PEAR odczytuje się za pomocą polecenia `pear help`.

Aby wykorzystać moduł PEAR w PHP, można użyć w kodzie skryptu instrukcję `require`. Użycie poniższej instrukcji włącza obsługę modułu DB:

```
require 'DB.php';
```

Można również zastosować instrukcję `include 'DB.php';`, która jednak różni się nieco od poprzedniej. Jeśli plik, który chcemy wykorzystać nie istnieje, instrukcja `require` uzna to za błąd krytyczny i zakończy skrypt, natomiast instrukcja `include` wyświetli jedynie ostrzeżenie. Moduł DB wykorzystamy w praktyce w następnym punkcie.

Wyświetlanie zawartości tabeli za pomocą modułu DB

Oto przykład użycia funkcji modułu DB do odczytania zawartości tabeli bazy danych MySQL. Najpierw należy włączyć obsługę modułu DB za pomocą instrukcji `require`:

```
require 'DB.php';
    .
    .
    .
```

Następnie należy podłączyć się do bazy danych za pomocą metody `DB::connect` (jak pamiętamy, za pomocą operatora `::` można wskazać klasę, do której należy metoda). Ogólny sposób wykorzystania tej metody przedstawia się następująco:

```
DB:connect(nazwatypudb://nazwauzytkownika:hasło@serwer/nazwabd);
```

Jeśli pracujemy na komputerze lokalnym, powinniśmy zastosować *localhost* zamiast nazwy serwera. Na przykład, aby połączyć się lokalnie z bazą danych `plody_rolne` z nazwą użytkownika `root` i bez hasła, należy wprowadzić następujący kod:

```
require 'DB.php';
$db = DB::connect("mysql://root:@localhost/plody_rolne");
    .
    .
    .
```

Teraz można użyć metody `query` obiektu `$db` i wykonać zapytanie SQL tak, jak poniżej, gdzie przeczytano całą zawartość tabeli `owoc`:

```
require 'DB.php';
$db = DB::connect('mysql://root:@localhost/plody_rolne');

$query = "SELECT * FROM owoc";

$result = $db->query($query);
```

Wiersz danych można pobrać za pomocą metody `fetchRow`. Przykład pokazano w skrypcie *phpdb.php*, zamieszczonym na listingu 8.8 — dzięki użyciu stałej `DB_FETCHMODE_ASSOC` metoda zwróciła dane w postaci tablicy asocjacyjnej. Pozwoliło to, by pozostała część kodu nie różniła się od odczytania zawartości tablicy z wykorzystaniem funkcji obsługi bazy MySQL.

Listing 8.8. *Wyświetlanie zawartości tabeli z wykorzystaniem modułu DB, phpdb.php*

```
<HTML>
    <HEAD>
        <TITLE>
            Zastosowanie modułu DB do wyświetlenia zawartości tabeli
        </TITLE>
    </HEAD>
    <BODY>
        <CENTER>
            <H1>Zastosowanie modułu DB do wyświetlenia zawartości tabeli</H1>
            <?php

                require 'DB.php';

                $db = DB::connect('mysql://root:@localhost/plody_rolne');

                $query = "SELECT * FROM owoc";

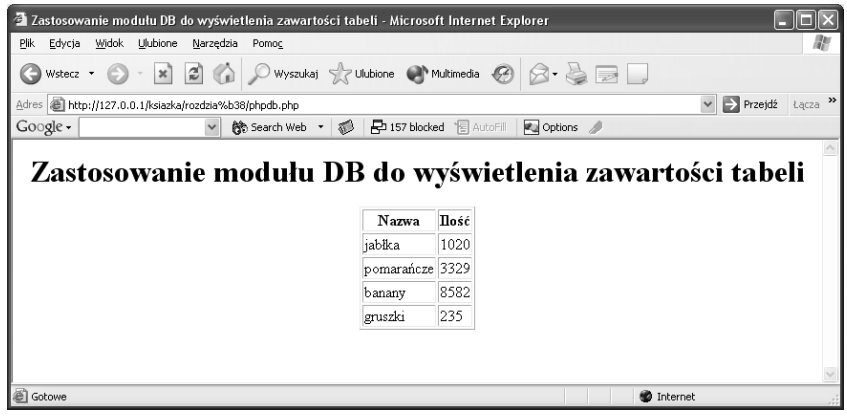
                $result = $db->query($query);

                echo "<TABLE BORDER='1'>";
                echo "<TR>";
                echo "<TH>Nazwa</TH><TH>Ilość</TH>";
                echo "</TR>":

                while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
                {
                    echo "<TR>";
                    echo "<TD>", $row['nazwa'], "</TD><TD>", $row['ilosc'],
                        "</TD>";
                    echo "</TR>";
                }
                echo "</TABLE>";

            ?>
        </CENTER>
    </BODY>
</HTML>
```

Wynik działania skryptu pokazano na rysunku 8.8.



Rysunek 8.8. *Wykorzystanie modułu DB do wyświetlenia tabeli*

Wprowadzanie nowych danych do bazy za pomocą modułu DB

Zobaczmy, jak za pomocą modułu DB wprowadza się do bazy nowe dane. W kolejnym przykładzie wprowadzimy dane o morelach tak, jak wcześniej zrobiliśmy to przy wykorzystaniu funkcji obsługi bazy danych MySQL. Zadanie jest proste. Najpierw trzeba się podłączyć do bazy danych za pomocą metody `DB::connect`:

```
$db = DB::connect('mysql://root:@localhost/plody_rolne');
    .
    .
    .
```

Poniższa instrukcja SQL pozwoli na wprowadzenie do tabeli `owoc` wiersza danych dotyczącego moreli:

```
$db = DB::connect('mysql://root:@localhost/plody_rolne');

$query = "INSERT INTO owoc (nazwa, ilosc) VALUES('morele', '203')";
    .
    .
    .
```

Teraz można wykonać zapytanie za pomocą metody `$db->query`:

```
$db = DB::connect('mysql://root:@localhost/plody_rolne');

$query = "INSERT INTO owoc (nazwa, ilosc) VALUES('morele', '203')";

$result = $db->query($query);
```

Po wprowadzeniu nowego wiersza wyświetlimy zawartość całej tabeli tak, jak pokazano w skrypcie *phpdbinsert.php*, na listingu 8.9.

Listing 8.9. *Wprowadzanie nowych danych do bazy, phpdbinsert.php*

```
<HTML>
    <HEAD>
        <TITLE>
            Wykorzystanie modułu DB do wprowadzania danych
        </TITLE>
    </HEAD>
    <BODY>
        <CENTER>
            <H1>Zastosowanie modułu DB do wprowadzania danych</H1>

            <?php
            require 'DB.php';

            $db = DB::connect('mysql://root:@localhost/plody_rolne');

            $query = "INSERT INTO owoc (nazwa, ilosc) VALUES('morele', '203')";

            $result = $db->query($query);

            $query = "SELECT * FROM owoc";

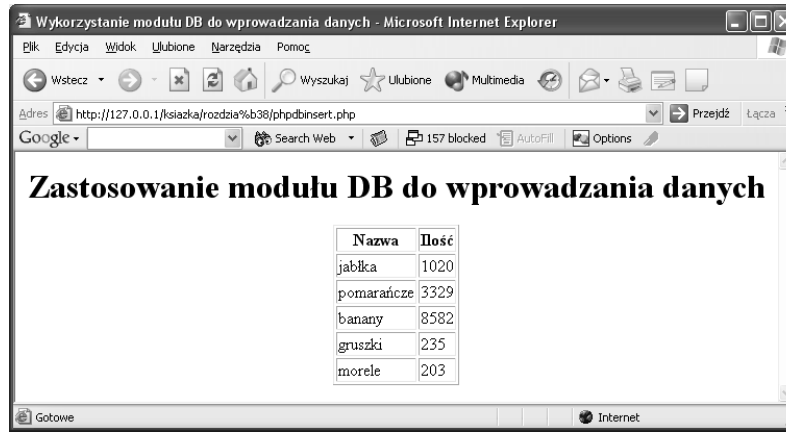
            $result = $db->query($query);

            echo "<TABLE BORDER='1'>";
            echo "<TR>";
            echo "<TH>Nazwa</TH><TH>Ilość</TH>";
            echo "</TR>";

            while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
            {
                echo "<TR>";
                echo "<TD>", $row['nazwa'], "</TD><TD>",
                             $row['ilosc'], "</TD>";
                echo "</TR>";
            }
            echo "</TABLE>";

        ?>
        </CENTER>
    </BODY>
</HTML>
```

Udało się. Planowany rekord dotyczący moreli został dodany, co można zobaczyć na rysunku 8.9.



Rysunek 8.9. *Wykorzystanie modułu DB do wprowadzania danych*

Aktualizacja danych za pomocą modułu DB

Oto kolejna operacja: aktualizowanie danych z wykorzystaniem modułu DB. Podobnie, jak poprzednio zmniejszymy ilość gruszek w tabeli owoc z 235 do 234 kilogramów. Mając już trochę doświadczenia w używaniu języka SQL z modułem DB, nie powinniśmy mieć z tym problemów. Wystarczy skonfigurować właściwą instrukcję SQL, a następnie ją wykonać:

```
$query = "UPDATE owoc SET ilosc = 234 WHERE nazwa = 'gruszki'";
$result = $db->query($query);
```

Przykładową aktualizację zaprezentowano w skrypcie *phpupdate.php*, na listingu 8.10.

Listing 8.10. *Aktualizacja danych w bazie, phpdbupdate.php*

```
<HTML>
    <HEAD>
        <TITLE>
            Wykorzystanie modułu DB do aktualizowania danych
        </TITLE>
    </HEAD>
    <BODY>
        <CENTER>
            <H1>Wykorzystanie modułu DB do aktualizowania danych</H1>

            <?php
                require 'DB.php';

                $db = DB::connect('mysql://root:@localhost/plody_rolne');

                $query = "UPDATE owoc SET ilosc = 234 WHERE nazwa = 'gruszki'";
```

```
                $result = $db->query($query);

                $query = "SELECT * FROM owoc";

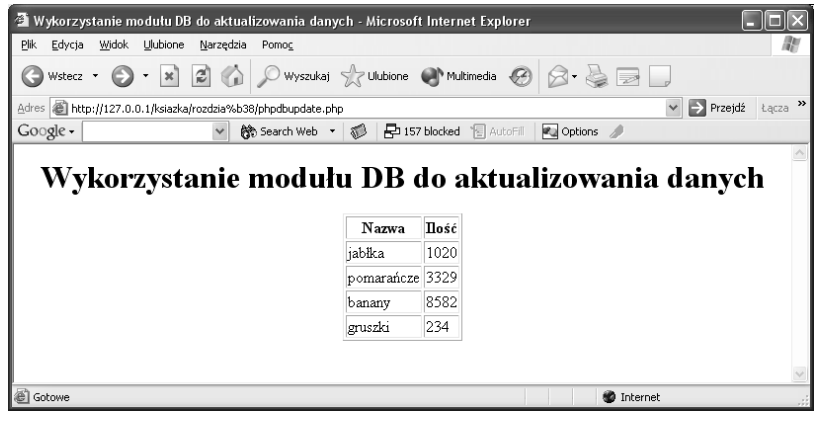
                $result = $db->query($query);

                echo "<TABLE BORDER='1'>";
                echo "<TR>";
                echo "<TH>Nazwa</TH><TH>Ilość</TH>";
                echo "</TR>";

                while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
                {
                    echo "</TR>";
                    echo "<TD>", $row['nazwa'], "</TD><TD>",
                                $row['ilosc'], "</TD>";
                    echo "</TR>";
                }
                    echo "</TABLE>";

            ?>
        </CENTER>
    </BODY>
</HTML>
```

Wyniki zaprezentowano na rysunku 8.10. Jak można zauważyć, bez trudu udało się zmodyfikować dane o ilości gruszek zapisane w bazie.



Rysunek 8.10. *Wykorzystanie modułu DB do aktualizacji danych*

W module DB jest również metoda `DB::isError`, która służy do obsługi błędów. Moduł DB, po uzyskaniu wyniku działania jego metody, można przekazać do metody `DB::isError`. Jeśli metoda ta zwróci `TRUE`, oznacza to, że wystąpił błąd. Właściwy komunikat o błędzie można wyświetlić za pomocą metody `getMessage` obiektu reprezentującego wynik:

```
$db = DB::connect('mysql://root:@localhost/plody_rolne');

if(DB::isError($db)){
    die($db->getMessage());
}
```

Teraz, kiedy umiemy już wykonywać instrukcje SQL za pomocą modułu DB, możemy tworzyć dowolnie skomplikowane aplikacje bazodanowe. Zmiana typu serwera bazy danych sprowadza się do podania nazwy typu serwera w ciągu połączenia.

Podsumowanie

W tym rozdziale zaprezentowano zagadnienia związane z obsługą baz danych i opisano różne opcje obsługi baz danych dostępne w PHP. Jest to bardzo przydatne, ponieważ pozwala na zapisywanie danych na serwerze i zarządzanie nimi. Oto kilka najważniejszych zagadnień opisanych w tym rozdziale:

- Do serwerów baz danych można podłączyć się za pomocą funkcji PHP `mysql_connect`. Bazę danych wybieramy dzięki funkcji `mysql_select_db`.
- Tworzenie bazy danych umożliwia instrukcja `CREATE`.
- Instrukcja `CREATE` pozwala także tworzyć tabele i określać typ danych, które są w nich zapisane.
- Dane wprowadza się do bazy danych za pomocą instrukcji `INSERT`.
- Dane można pobierać z bazy danych, używając instrukcji `SELECT`.
- Aktualizację danych w tabeli bazy danych wykonuje się za pomocą instrukcji `UPDATE`.
- Działanie instrukcji `DELETE` usuwa dane z tabel bazy danych.