helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

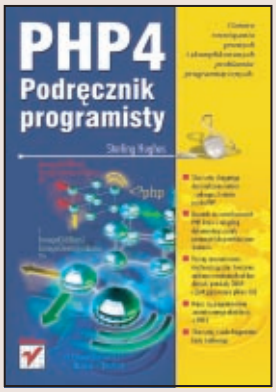
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

PHP4. Podręcznik programisty



Autor: Sterling Hughes
Tłumaczenie: Daniel Kaczmarek
ISBN: 83-7197-795-6
Tytuł oryginału: PHP Developer's Cookbook
Format: B5, stron: 422
Przykłady na ftp: 77 kB

Od teorii do praktyki prowadzi długa droga. Nie inaczej jest w przypadku PHP4, popularnego języka służącego do tworzenia aplikacji WWW. Oficjalna dokumentacja stanowi jedynie wstęp – w praktyce programista często staje wobec problemów, których rozwiązań nie zawiera sam opis języka.

Książka ta stanowi kombinację podręcznika i instrukcji. Przedstawia treści, które nie znalazły się w dokumentacji, a w wielu przypadkach znacznie tę dokumentację rozszerza. Pomimo tego, że nie jest to kompletna instrukcja, staniesz się po jej lekturze dużo lepszym programistą – będziesz miał solidne podstawy do rozwiązywania problemów programistycznych.

Opisano w niej:

- Operacje na łańcuchach znaków
- Pracę z liczbami, datami i czasem
- Użycie tablic w PHP
- Wbudowane tablice i stałe PHP
- Wyrażenia regularne
- Pracę z plikami i katalogami
- Tworzenie własnych funkcji
- Klasy i obiekty w PHP
- Sesje
- Pisanie aplikacji niezależnych od baz danych
- Łączenie kodu PHP z programami napisanymi w innych językach
- Obsługę poczty elektronicznej
- Użycie protokołów SNMP i LDAP
- Tworzenie grafiki w PHP
- Pracę z dokumentami XML

„Sterling i Andrei stworzyli podręcznik zawierający rozwiązania konkretnych problemów, przedstawiając jednocześnie informacje niezbędne do rozwiązania niemal każdego problemu”. – Rasmus Lerdorf, twórca PHP

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

Spis treści

O Autorach **13**
Przedmowa **15**
Wprowadzenie **19**

Część I Konstrukcje i techniki językowe **25**

Rozdział 1. Operacje na łańcuchach znaków **27**
1.0. Wprowadzenie 27
1.1. Rozkładanie łańcuchów znaków 27
1.2. Stosowanie operatora trójargumentowego 31
1.3. Wymiana wartości zmiennych 32
1.4. Konwersja kodów ASCII w znaki 33
1.5. Dzielenie łańcucha na pojedyncze znaki 34
1.6. Odwracanie kolejności we fragmentach łańcuchów 36
1.7. Zmiana wielkości liter w łańcuchu 37
1.8. Usuwanie pustych przestrzeni z łańcucha znaków 38
1.9. Ucieczka od znaków specjalnych w łańcuchach 39
1.10. Czytanie z pliku z tekstem dzielonym przecinkami 40
1.11. Parsowanie adresu URL 42
1.12. Dopasowywanie przybliżone 42
1.13. Tworzenie unikalnego identyfikatora 43
1.14. Szyfrowanie łańcucha znaków 44
1.15. Konwersja między zestawami znaków cyrylicy 46

Rozdział 2. Praca z liczbami, datami i czasem **49**
2.0. Wprowadzenie 49
2.1. Sprawdzanie, czy zmienna jest poprawną liczbą 50
2.2. Praca z ciągami liczbowymi 51
2.3. Praca z liczbami nie będącymi liczbami całkowitymi ani zmiennoprzecinkowymi 52
2.4. Zaokrąglanie liczb o stałej precyzji 54
2.5. Przekształcanie liczb o różnych bazach 55
2.6. Obliczanie logarytmu liczby 56
2.7. Znajdowanie binarnej reprezentacji liczby 57
2.8. Konwersja między liczbami arabskimi i rzymskimi 58
2.9. Weryfikacja poprawności numerów kart kredytowych 58
2.10. Formatowanie liczb 60
2.11. Konwersja między radianami i stopniami 61
2.12. Obliczanie cosinusów, sinusów i tangensów 62
2.13. Generowanie liczb losowych 63
2.14. Generowanie unikalnych liczb losowych 64
2.15. Ważenie liczb losowych 67

2.16. Ładowanie dzisiejszej daty do tablicy ... 68
2.17. Weryfikacja poprawności daty ... 69
2.18. Obliczanie odstępów czasowych ... 70
2.19. Znajdowanie daty i godziny w innych regionach ... 71
2.20. Formatowanie znaczników czasu ... 72
2.21. Odczytywanie daty i godziny z łańcuchów znaków ... 75
2.22. Przeprowadzanie testów wydajności ... 77
2.23. Zatrzymywanie wykonania programu ... 81

Rozdział 3. Stosowanie tablic ... 83
3.0. Wprowadzenie ... 83
3.1. Deklarowanie tablicy ... 84
3.2. Wypisywanie zawartości tablicy ... 85
3.3. Usuwanie powtarzających się elementów ... 86
3.4. Powiększanie lub pomniejszanie tablicy ... 87
3.5. Łączenie tablic ... 88
3.6. Iteracyjne przetwarzanie elementów tablicy ... 89
3.7. Uzyskiwanie dostępu do bieżącego elementu tablicy ... 92
3.8. Uzyskiwanie dostępu do różnych obszarów tablicy ... 93
3.9. Przeszukiwanie tablicy ... 94
3.10. Odnajdywanie elementów różniących dwie tablice ... 95
3.11. Losowe ustawianie elementów w tablicy ... 96
3.12. Wyznaczanie iloczynu, różnicy i punktu przecięcia dwóch tablic ... 97
3.13. Sortowanie tablicy ... 98
3.14. Sortowanie naturalne ... 100
3.15. Odwracanie kolejności ... 100
3.16. Funkcje tablicowe analogiczne do języka Perl ... 101

Rozdział 4. Wbudowane tablice i stałe PHP ... 103
4.0. Wprowadzenie ... 103
4.1. Wykorzystanie stałych plikowych ... 104
4.2. Stałe wersji PHP i systemu operacyjnego ... 105
4.3. Ustawianie punktów kontrolnych za pomocą stałych błędów PHP ... 106
4.4. Definiowanie własnych stałych PHP ... 107
4.5. Wykorzystanie zmiennych globalnych PHP ... 108
4.6. Uzyskiwanie dostępu do danych za pośrednictwem wbudowanych tablic PHP ... 110

Rozdział 5. Dopasowywanie danych do wyrażeń regularnych ... 113
5.0. Wprowadzenie ... 113
5.1. Przypisywanie wyników zastąpienia wzorca ... 120
5.2. Stosowanie w PHP wyrażeń regularnych zgodnych z Perlem ... 121
5.3. Niezgodności między biblioteką PCRE a wyrażeniami regularnymi Perla ... 123
5.4. Dopasowywanie wielu wierszy ... 125
5.5. Znajdowanie wystąpienia dopasowania o określonym numerze ... 126
5.6. Praca z separowanymi rekordami ... 128
5.7. Wyodrębnianie określonych wierszy ... 129
5.8. Sprawdzanie znaków ... 130
5.9. Weryfikacja poprawności danych WWW ... 132
5.10. Weryfikacja poprawności adresu poczty elektronicznej ... 133
5.11. Sprawdzanie składni wyrażenia regularnego ... 133
5.12. Sprawdzanie powtórzeń słów ... 134
5.13. Skracanie danych wejściowych ... 135

Rozdział 6. Obsługa plików 137
6.0. Wprowadzenie 137
6.1. Sprawdzanie, czy dany plik istnieje 139
6.2. Sprawdzanie praw do pliku 140
6.3. Tworzenie pliku tymczasowego 141
6.4. Przechowywanie pliku wewnątrz programu 142
6.5. Otwieranie pliku 143
6.6. Bezpieczna obsługa danych binarnych 144
6.7. Opróżnianie bufora 145
6.8. Blokowanie plików 146
6.9. Sprawdzanie ilości wolnego miejsca na wskazanym napędzie 148
6.10. Wyświetlanie zawartości pliku tekstowego użytkownikowi 149
6.11. Operacje na standardowych strumieniach wejścia-wyjścia 150
6.12. Czytanie z pliku wiersz po wierszu 151
6.13. Przetwarzanie pliku słowo po słowie 152
6.14. Przetwarzanie pliku w odwrotnej kolejności 153
6.15. Parsowanie pliku na podstawie wzorcowych separatorów 154
6.16. Zmiana wartości określonego rekordu 155
6.17. Uzyskiwanie dostępu do rekordów o stałej długości 156
6.18. Odczytywanie z pliku pojedynczego wiersza 157
6.19. Przycinanie pliku 158
6.20. Zliczanie wierszy w pliku 159
6.21. Odczytywanie z pliku losowo wybranego wiersza 160
6.22. Ustawianie wierszy i słów w porządku losowym 161
6.23. Tworzenie plików konfiguracyjnych 163

Rozdział 7. Praca z plikami w katalogach 165
7.0. Wprowadzenie 165
7.1. Praca ze znacznikami czasu 166
7.2. Usuwanie pliku 166
7.3. Kopiowanie lub przenoszenie pliku 167
7.4. Śledzenie nazw plików 168
7.5. Parsowanie części nazwy pliku 169
7.6. Ładowanie wszystkich plików w katalogu do tablicy 170
7.7. Przeszukiwanie systemu plików 170
7.8. Przetwarzanie katalogów plik po pliku 172
7.9. Rekurencyjne usuwanie katalogu 172
7.10. Tworzenie wyszukiwarki 174

Rozdział 8. Funkcje 181
8.0. Wprowadzenie 181
8.1. Przekazywanie do funkcji wartości domyślnej 182
8.2. Dostęp do zmiennych spoza funkcji 183
8.3. Zwracanie wartości przez funkcję 184
8.4. Przekazywanie argumentów przez odwołanie 185
8.5. Zachowywanie wartości zmiennej między kolejnymi wywołaniami funkcji 187
8.6. Zwracanie przez funkcję więcej niż jednej wartości 187
8.7. Dynamiczne deklarowanie funkcji 188
8.8. Dynamiczne tworzenie funkcji anonimowych 189
8.9. Pośrednie wywoływanie funkcji 190
8.10. Pobieranie arbitralnie określonej liczby parametrów 190

Rozdział 9. Klasy 193
9.0. Wprowadzenie 193
9.1. Tworzenie klasy 194

9.2. Uzyskiwanie dostępu do zmiennych z wnętrza klasy ... 195
9.3. Dziedziczenie ... 196
9.4. Tworzenie zmiennych lub funkcji publicznych lub prywatnych ... 198
9.5. Tworzenie konstruktora ... 199
9.6. Zwracanie innego obiektu przez konstruktor ... 200
9.7. Tworzenie destruktora klasy ... 201
9.8. Używanie funkcji klasy bez inicjowania obiektu ... 202
9.9. Pośredni dostęp do metod klasy rodzica ... 202
9.10. Zwracanie obiektu błędu w razie niepowodzenia ... 204

Rozdział 10. Utrzymywanie sesji w PHP ... 207
10.0. Wprowadzenie ... 207
10.1. Tworzenie zmiennej sesji w PHP ... 208
10.2. Zapisywanie sesji przy użyciu bazy danych ... 210
10.3. Ustawianie nazwy sesji ... 213
10.4. Ustawianie i pobieranie parametrów cookie ... 214
10.5. Wyrejestrowywanie zmiennej z sesji ... 215
10.6. Usuwanie wszystkich zmiennych sesji ... 215
10.7. Stosowanie obiektu jako zmiennej sesji ... 216
10.8. Kodowanie danych ... 218
10.9. Tworzenie koszyka na zakupy z zastosowaniem sesji w PHP ... 218
10.10. Serializacja ... 224
10.11. Serializacja WDDX ... 225
10.12. Deserializacja WDDX ... 226

Rozdział 11. Interakcja z witrynami i serwerami WWW ... 227
11.0. Wprowadzenie ... 227
11.1. Pobieranie strony WWW ... 227
11.2. Przeprowadzanie transakcji SSL ... 228
11.3. Przeprowadzanie wywołania HTTP POST ... 230
11.4. Załadowywanie pliku w protokole HTTP ... 231
11.5. Wysyłanie cookies w ramach wywołania ... 233
11.6. Dołączanie lub pomijanie nagłówków z transmisji cURL ... 234
11.7. Łączenie się za pośrednictwem serwera proxy ... 235
11.8. Pobieranie informacji dotyczących transmisji cURL ... 236
11.9. Interakcja z ramkami ... 237
11.10. Uzyskiwanie wszystkich URL ze strony WWW ... 238
11.11. Wyszukiwanie łączy pustych i prawidłowych ... 239
11.12. Uzyskiwanie nowych łączy ze strony WWW ... 240
11.13. Tworzenie lustrzanej kopii strony WWW ... 242
11.14. Parsowanie i formatowanie pliku dziennika zdarzeń ... 242

Część II Bazy danych ... 245

Rozdział 12. Tworzenie API niezależnego od baz danych w PHP ... 247
12.0. Wprowadzenie ... 247
12.1. Łącznik ... 249
12.2. Moduł MySQL ... 251
12.3. Moduł mSQL ... 254
12.4. Moduł Oracle ... 256
12.5. Moduł MSSQL ... 258
12.6. Moduł ODBC ... 260
12.7. Moduł PostgreSQL ... 261
12.8. Moduł InterBase ... 263
12.9. Moduł Sybase ... 265

Część III Wychodzenie na zewnątrz PHP....267

Rozdział 13. Łączenie z innymi programami i językami 269
13.0. Wprowadzenie....269
13.1. Przechwytywanie danych wynikowych innego programu....270
13.2. Wyświetlanie danych wynikowych programu....271
13.3. Otwieranie potoku do innego programu....271
13.4. Praca z gniazdami....272
13.5. Praca z obiektami COM....273
13.6. Uzyskiwanie dostępu do predefiniowanych metod i klas języka Java....275
13.7. Uzyskiwanie dostępu do własnych klas i metod języka Java....276

Rozdział 14. Komunikacja z gniazdami 281
14.0. Wprowadzenie....281
14.1. Klient TCP....281
14.2. Serwer TCP....283
14.3. Czytanie z gniazd i zapisywanie do nich....284
14.4. Klient UDP....285
14.5. Serwer UDP....287
14.6. Domeny gniazd Uniksa....288
14.7. Obsługa wielu adresów IP....290
14.8. Gniazda nieblokujące się....290
14.9. Odczytywanie i zapisywanie wektorów wejścia-wyjścia....291
14.10. Sterowanie czasem transmisji danych....292
14.11. Sprawdzanie statusu gniazda....293

Rozdział 15. Obsługa wiadomości poczty elektronicznej 295
15.0. Wprowadzenie....295
15.1. Otwieranie skrzynki pocztowej IMAP....296
15.2. Sprawdzanie, czy strumień IMAP jest nadal aktywny....297
15.3. Przekształcanie wiadomości do czytelnej postaci....298
15.4. Wysyłanie wiadomości poczty elektronicznej....298
15.5. Wysyłanie załączników przy użyciu PHP....299
15.6. Wysyłanie załączników binarnych....300
15.7. Wysyłanie wiadomości w formacie HTML....303
15.8. Sprawdzanie rozmiaru wiadomości....305
15.9. Parsowanie nagłówków wiadomości....305

Rozdział 16. Praca z obiektami SNMP.... 307
16.0. Wprowadzenie....307
16.1. Ustawianie obiektu SNMP....308
16.2. Pobieranie obiektu SNMP....308
16.3. Pobieranie wszystkich obiektów SNMP do tablicy....309

Rozdział 17. LDAP.... 311
17.0. Wprowadzenie....311
17.1. Dodawanie pozycji do serwera LDAP....311
17.2. Usuwanie pozycji z serwera LDAP....312
17.3. Wykonywanie zapytania i pobieranie wyników....313
17.4. Zwalnianie zbioru wyników LDAP....314
17.5. Przeszukiwanie drzewa....315
17.6. Sortowanie wyników przeszukiwania....317

Część IV Generowanie innych języków 319

Rozdział 18. Tworzenie obrazków i zarządzanie nimi 321
18.1. Tworzenie obrazka z GD 322
18.2. Otwieranie obrazka już istniejącego 323
18.3. Sprawdzanie rozmiaru obrazka 324
18.4. Dodawanie tekstu do obrazków 325
18.5. Sprawdzanie koloru określonej części obrazka 329
18.6. Sprawdzanie całkowitej liczby kolorów w obrazku 330
18.7. Tworzenie przezroczystych obrazków GIF/PNG 331
18.8. Kopiowanie części jednego obrazka do drugiego 332
18.9. Rysowanie prostokątów 333
18.10. Rysowanie wielokątów 334
18.11. Rysowanie łuku 335
18.12. Tworzenie obrazka z przeplotem 337
18.13. Przyciski dynamiczne 337
18.14. Stosowanie czcionek TrueType 340

Rozdział 19. HTML 341
19.0. Wprowadzenie 341
19.1. Usuwanie znaczników HTML 341
19.2. Konwersja ASCII do HTML 342
19.3. Generowanie list <select> 343
19.4. Generowanie skryptu JavaScript realizującego efekty rollover 345
19.5. Tworzenie szablonów HTML 346

Rozdział 20. XML 349
20.0. Wprowadzenie 349
20.1. Obsługa błędów 351
20.2. Parsowanie prostych dokumentów XML 352
20.3. Parsowanie dokumentu XML do tablicy 354
20.4. Odwzorowywanie znaczników XML 356
20.5. Ustawianie uchwytu zewnętrznego odwołania do jednostki 357
20.6. Przeszukiwanie XML 361
20.7. Oszczędzanie pamięci 363
20.8. Ustawianie i sprawdzanie opcji 363
20.9. Parsowanie z zastosowaniem funkcji DOM-XML 364
20.10. Tworzenie dokumentu XML 367
20.11. Przekształcanie XML za pomocą XSL 369
20.12. Filtrowanie danych wyjściowych za pomocą pliku XSL 370

Część V API Zend 371

Rozdział 21. API Zend 373
21.0. Wprowadzenie 373
21.1. Pobieranie argumentów 374
21.2. Modyfikowanie argumentów funkcji 376
21.3. Zwracanie łańcuchów znaków lub liczb przez funkcję 378
21.4. Zwracanie tablic i obiektów przez funkcję 379
21.5. Dodawanie funkcji do PHP 381
21.6. Tworzenie identyfikatorów zasobów 382
21.7. Pobieranie identyfikatorów zasobów 383
21.8. Przechodzenie w pętli przez tablice 384
21.9. Tworzenie modułu PHP 386
21.10. Dodawanie własnego pliku do instalacji PHP 400

Dodatki 403

Dodatek A Instalacja PHP 405

Dodatek B Identyfikowanie i usuwanie usterek w PHP 411
Najczęstsze błędy i co one oznaczają 411
Techniki unikania błędów i usterek 413

Dodatek C Zasoby online PHP 415
Oficjalna witryna PHP 415
Witryna Zend 415
PHPBuilder 416
PHPWizard.net 416
Repozytorium klas PHP 416
Weberdev 416
DevShed 417

Dodatek D Migracja do PHP4 419
Inicjujące zmienne statyczne i argumenty domyślne akceptują tylko wartości skalarne 419
Instrukcje break i continue mają zasięg lokalny względem plików dołączanych lub łańcuchów znaków przetwarzanych funkcją eval 420
Instrukcja return z pliku dołączanego instrukcją require nie działa 420
Unset jest teraz instrukcją, a nie funkcją 421
"{$" nie jest obsługiwane w łańcuchach znaków 421

Skorowidz 423

Rozdział 10.

Utrzymywanie sesji w PHP

„Prawdziwą sztuką pamięci jest sztuka skupiania uwagi”.

Samuel Johnson

W rozdziale tym poruszone zostaną zagadnienia związane z utrzymywaniem stanu — procesem polegającym na zachowywaniu wartości zmiennych między wywołaniami kolejnych skryptów. Najpierw omówione zostaną nowe mechanizmy zarządzania sesjami w PHP4, po czym przejdziemy do części poświęconej niewielkiemu systemowi koszyków na zakupy. Następnie przyjrzymy się funkcjom serializacji w PHP (w tym WDDX), które stanowią wydajny sposób przechowywania zmiennych PHP.

10.0. Wprowadzenie

HTTP jest protokołem bezstanowym, co oznacza, że po opuszczeniu przez użytkownika strony WWW lub zakończeniu działania aplikacji komputer utraci wszelkie dane dotyczące przeprowadzonych transakcji (chyba że oprzesz się na dziennikach zdarzeń systemowych serwera WWW). Począwszy od Netscape 3.0, firma Netscape rozwiązywała ten problem za pomocą *cookies*. *Cookies* to pliki przechowywane na komputerze użytkownika, dostępne dla skryptu, który je wysłał. Umożliwiają one zapisywanie i odczytywanie informacji o ewentualnych wcześniejszych wizytach, które użytkownik odbył na stronie WWW.

Łącząc *cookies* i bazy danych, możesz zapisywać na swoim komputerze każdy bit informacji posiadanych na temat użytkownika oraz odczytywać je, stosując unikalny identyfikator przypisany temu użytkownikowi, przechowywany w *cookie* na jego komputerze. Podejście takie wymagało jednak znacznego nakładu pracy programistów, a uruchomienie zabierało dużo czasu. Dlatego aby ułatwić Ci życie, grupa zajmująca się rozwojem PHP udostępniła ściśle zintegrowany zbiór narzędzi służących do zarządzania sesjami.

To, co w każdym przypadku wymagało dotychczas od dwudziestu do stu wierszy kodu, teraz zajmuje co najwyżej kilka wierszy dobrze przemyślanego kodu źródłowego.

Jak to działa?

Sesje działają w następujący sposób. W wyniku wywołania funkcji `session_start()` lub `session_register()` PHP ładuje ze schowka sesji zapisane dane dotyczące sesji. Podczas wykonywania skryptu możesz za pomocą funkcji `session_register()` rejestrować dane, które mają zostać zapisane w schowku sesji. W momencie zakończenia wykonywania skryptu PHP zmienne sesji zostają zapisane w schowku sesji, którego ścieżka wskazywana jest przez pozycję `session.save_path` w pliku *php.ini*. Przypisany sesji unikalny identyfikator jest wstawiany do pliku *cookie* i wysyłany do użytkownika, ale to dzieje się tylko przy pierwszym uruchomieniu sesji.

Czasami serializacja

Jednym ze sposobów utrzymywania stanu jest zastosowanie modułu sesji. Jednak w niektórych przypadkach nie będziesz zmuszony korzystać ze wszystkich możliwości tego modułu — wystarczy zapisywanie i odczytywanie pewnych zmiennych. W takiej sytuacji możesz zastosować udostępniane przez PHP funkcje serializacji (`serialize()` i `unserialize()`) lub skorzystać z dołączonego rozszerzenia *WDDX*, które zapisuje zmienne PHP w standardowym formacie, przez co mogą one być później odczytywane i wykorzystywane przez inne języki.

10.1. Tworzenie zmiennej sesji w PHP

Chcesz utworzyć zmienną sesji przy zastosowaniu PHP.

Technika

Aby zarejestrować zmienną sesji, zastosuj funkcję `session_register()`:

```
<?php
session_register('session_variable');

$session_variable = $session_variable ? $session_variable + 1 : 20;

echo $session_variable;
?>
```

Uwagi

W PHP istnieją obecnie trzy typy zmiennych o różnym zasięgu i spełniające różne cele. Pierwszym typem zmiennej, mającym najmniejszy zasięg, jest zmienna lokalna. Zmienne

lokalna to każda zmienna znajdująca się w zasięgu funkcji i istniejąca tylko na czas wykonania tej funkcji. Przeanalizujmy następujący przykład:

```
<?php
srand((double)microtime() * 1000000);

function get_number($num) {
    $num *= rand();
    return ($num);
}

$number1 = get_number(rand());
$number2 = get_number(5);
$number3 = get_number(5);

print "Liczba numer 1 to $number1\n<br>\n
       Liczba numer 2 to $number2\n<br>\n
       Liczba numer 3 to $number3\n<br>\n";
?>
```

W powyższym przykładzie zmienna `$num` ma zasięg lokalny, a więc jej wartość nie jest dostępna spoza zasięgu funkcji `get_number()`.

Następnym typem zmiennej jest zmienna globalna, czyli każda zmienna zadeklarowana poza klasą lub funkcją (lub zadeklarowana instrukcją `global` albo za pośrednictwem tablicy `$GLOBALS`). Zmienne globalne istnieją przez cały czas trwania skryptu.

Ostatnim typem zmiennej wprowadzonym w PHP4 jest zmienna sesji. Zmienna sesji może (teoretycznie) trwać wiecznie, chyba że Ty lub użytkownik ją usuniecie (dzięki odpowiedniemu ustawieniu opcji konfiguracyjnych zmienne takie mogą również być automatycznie usuwane po upływie pewnego czasu).

Tak więc zmienne sesji to po prostu zwykłe zmienne zadeklarowane przy użyciu funkcji `session_register()`. Zmienne te mogą być dowolnego typu obsługiwanego przez PHP: tablicą, łańcuchem znaków, liczbą, a nawet obiektem. Zachowanie `session_register()` zależy od wartości parametru konfiguracyjnego `register_globals`. Jeśli wartością `register_globals` jest `on`, `session_register('foo')` zapisze zmienną globalną `$foo` w schowku sesji. Jeżeli natomiast ma on wartość `off`, wykorzystana zostanie tablica `$HTTP_SESSION_VARS[]` i nastąpi zapisanie `$HTTP_SESSION_VARS['foo']`.

Podczas lektury tego rozdziału pamiętaj o poruszonych tu kwestiach. W dalszej kolejności będziemy zajmować się takimi kwestiami, jak:

- Zapisywanie zmiennych sesji w bazie danych.
- Utrzymywanie sesji przeglądarki między odsłonami kolejnych stron WWW.
- Ustawianie i pobieranie nazw sesji.
- Usuwanie zmiennych sesji.
- Znajdowanie ścieżki, w której zapisany jest identyfikator sesji.

10.2. Zapisywanie sesji przy użyciu bazy danych

Wolisz zapisywać dane sesji w bazie danych, a nie w systemie plików.

Technika

Zastosuj funkcję `session_set_save_handler()`, aby zarejestrować funkcje operujące na bazie danych:

Skrypt 10.1.

```
<?php
//
// schemat tabeli 'sessions'
// create table sessions (
//   session_id char(32) not null,
//   session_data text not null,
//   session_expiration int(11) unsigned not null,
//   primary key (session_id));
//

include_once 'DB.php';

// Zmienne globalne
$dbh = NULL;

function on_session_start ($save_path, $session_name) {
    global $dbh;

    $dbh = DB::connect('mysql://user:secret@localhost/SITE_SESSIONS', true);

    if (DB::isError($dbh)) {
        die(sprintf('Błąd [%d]: %s', $dbh->getCode(), $dbh->getMessage()));
    }
}

function on_session_end ()
{
    // Ta funkcja nie musi nic robić,
    // bo użyliśmy połączenia stałego
}

function on_session_read ($key)
{
    global $dbh;

    $stmt = "select session_data from sessions";
    $stmt .= " where session_id = '$key'";
    $stmt .= " and session_expiration > now()";
```

```
    $sth = $dbh->query($sth);
    $row = $sth->fetchRow(DB_FETCHMODE_ASSOC);
    return $row['session_data'];
}

function on_session_write ($key, $val)
{
    global $dbh;

    $val = addslashes($val);

    $insert_stmt = "insert into sessions values('$key', '$val', now() + 3600)";
    $update_stmt = "update sessions set session_data = '$val', ";
    $update_stmt .= "session_expiration = now() + 3600 ";
    $update_stmt .= "where session_id = '$key'";

    // Najpierw próbujemy polecenia insert; jeśli nie zadziała, oznacza to,
    // że sesja jest już zapisana w tabeli i trzeba ją uaktualnić
    if (DB::isError($dbh->query($insert_stmt)))
        $dbh->query($update_stmt);
}

function on_session_destroy ($key)
{
    global $dbh;

    $stmt = "delete from sessions where session_id = '$key'";
    $dbh->query($stmt);
}

function on_session_gc ($max_lifetime)
{
    global $dbh;

    // W tym przykładzie nie będziemy stosować parametru $max_lifetime
    // Po prostu usuniemy wszystkie sesje, które wygasły
    $stmt = "delete from sessions where session_expiration < now()";
    $dbh->query($stmt);
}

session_start();

// Zarejestruj zmienną $counter jako część sesji
session_register("counter");

// Wskaż funkcje przechowywania sesji
session_set_save_handler ("on_session_start",   "on_session_end",
                          "on_session_read",    "on_session_write",
                          "on_session_destroy", "on_session_gc");

// Sprawdźmy, co się stanie
$counter++;

print $counter;
```

```
session_destroy();

?>
```

Uwagi

Funkcja `session_set_save_handler()` umożliwia wskazanie funkcji obsługi, które będą wywoływane przez system sesji w celu wykonania operacji rozpoczęcia, zakończenia, załadowania i zapisania sesji.

Pierwszym argumentem `session_set_save_handler()` jest funkcja, która będzie wywoływana przez system sesji w momencie inicjacji sesji. Funkcja ta pobiera dwa argumenty: pierwszy to ścieżka, w której przechowywana była sesja (jest ona taka sama, jak wartość dyrektywy `session.save_path`), a drugi to nazwa ustawionego *cookie* sesji (domyślnie jest to `PHPSESSID`).

Następnym argumentem jest funkcja wywoływana w momencie zakończenia sesji (zazwyczaj na końcu wykonania skryptu) i powinna ona czyścić wszystkie dane. Funkcja ta nie pobiera żadnych argumentów.

Trzecim argumentem funkcji `session_set_save_handler()` jest funkcja wykonywana w momencie, gdy konieczne jest odczytanie danych sesji ze schowka sesji. Argumentem pobieranym przez tę funkcję jest identyfikator sesji (a więc otrzymuje ona coś w rodzaju `95f554d94b898773b604d0317773c68b`). *Identyfikator sesji* to losowa liczba wygenerowana, żeby utrudnić hakerom odgadnięcie identyfikatora sesji (i przez to uzyskanie dostępu do danych użytkownika).

Czwartym argumentem `session_set_save_handler()` jest funkcja wykonywana w momencie, gdy zachodzi konieczność zapisania danych w schowku sesji. Funkcja wskazywana przez ten argument pobiera wartość identyfikatora bieżącej sesji (patrz poprzedni akapit) oraz dane sesji mające postać pojedynczego łańcucha znaków.

Piątym argumentem `session_set_save_handler()` jest funkcja wykonywana w sytuacji, gdy wartość sesji ma ulec zniszczeniu. Funkcja ta pobiera argument, którego wartością jest identyfikator bieżącej sesji.

Szóstym i ostatnim argumentem `session_set_save_handler()` jest funkcja odśmiecania. Jest ona co jakiś czas wywoływana przez system sesji w celu usunięcia tych sesji, które już wygasły. Funkcja wskazywana przez ten argument pobiera liczbę sekund, po upłynięciu których dane sesji powinny zostać uznane za przestarzałe i usunięte.

Funkcja `session_set_save_handler()` stanowi podstawę przystosowywania mechanizmu zarządzania sesjami w PHP do własnych potrzeb. Choć może nie wydaje się to łatwe, pobaw się nią trochę. Wypróbuj jakieś trywialne skrypty wraz z `session_set_save_handler()`, a sam zobaczysz, jakie to proste.

10.3. Ustawianie nazwy sesji

Nazwa sesji to nazwa przechowywana przez przeglądarkę użytkownika i wyświetlana tym użytkownikom odwiedzającym stronę WWW, którzy mają ustawione ostrzeganie o *cookies*. Chcesz zmienić nazwę sesji z takiej, jaka określona jest w pliku konfiguracyjnym.

Technika

Aby zmienić nazwę *cookie* sesji, zastosuj funkcję `session_name()`. Pamiętaj, że należy ją wywoływać przed funkcjami `session_register()` i `session_start()`:

Skrypt 10.2.

```
<?php
$old_session = session_name('WebsiteTracker');
//
// wywołana bez parametrów session_name
// zwróci aktualną nazwę sesji
//
$new_session = session_name();

// zarejestruj nową zmienną sesji
session_register('session_variable');

print "Dotychczasowa nazwa sesji brzmiała: $old_session, ";
print "nowa nazwa sesji to $new_session";
?>
```

Uwagi

Domyślną wartością nazwy *cookie* przechowywanego przez przeglądarkę użytkownika jest `PHPSESSID`, lecz nie jest to nazwa zbyt wiele mówiąca. Tak naprawdę większość spośród użytkowników, którzy mają włączone ostrzeganie o *cookies* i zostaną poproszeni o zaakceptowanie *cookie* o nazwie `PHPSESSID`, najprawdopodobniej będzie się wahać. Funkcja `session_name()` pozwala na zmianę nazwy *cookie* w fazie wykonania, dzięki czemu nie musisz wprowadzać zmian na całej witrynie, modyfikując wartość parametru `session.name` w pliku *php.ini*.

Istnieje kilka zasad dotyczących wywoływania funkcji `session_name()`. Po pierwsze, nazwa sesji musi być alfanumeryczna — nie może zawierać znaków zapytania, tyld (~) i tak dalej. Po drugie, aby wywołanie funkcji `session_name()` przyniosło odpowiednie efekty, musi ona być wywoływana przed funkcjami `session_start()` i `session_register()`.

Wywołana z parametrem `session_name()` zmieni wartość zmiennej nazwy sesji i zwróci nazwę dotychczasową. Jeśli jednak `session_name()` zostanie wywołana w kontekście `void`, zwróci po prostu nazwę bieżącej sesji w postaci łańcucha znaków.

10.4. Ustawianie i pobieranie parametrów cookie

Chcesz pobierać lub ustawiać parametry *cookie* w fazie wykonania, a nie w pliku *php.ini*.

Technika

Aby ustawić parametry, zastosuj funkcję `session_set_cookie_params()`, która umożliwia ustawienie czasu życia *cookie*, jego ścieżki oraz domeny:

```
<?php
session_set_cookie_params(time()+8600, '/cookiepath', 'designmultimedia.com');
?>
```

Aby odczytać parametry *cookie*, zastosuj funkcję `session_get_cookie_params()`:

Skrypt 10.3.

```
<?php
$cookie_params = session_get_cookie_params();
print "Czas życia cookie: $cookie_params[lifetime]\n<br>\n
       Ścieżka cookie:    $cookie_params[path]\n<br>\n
       Domena cookie:     $cookie_params[domain]\n<br>\n";
?>
```

Uwagi

Funkcja `session_set_cookie_params()` ma następującą składnię:

```
int session_set_cookie_params(int czas_zycia [, string sciezka [, string domena]]);
```

Umożliwia ona modyfikowanie w fazie wykonania parametrów *cookie* określonych w pliku *php.ini*. Parametr `czas_zycia` określa, ile czasu ma upłynąć do wygaśnięcia *cookie*. Opcjonalny parametr `sciezka` wskazuje ścieżkę, w jakiej *cookie* będzie przechowywane, a `domena` domenę, która będzie udostępniać *cookie*. Funkcja zwraca wartość `1` w razie powodzenia i `0` w przypadku niepowodzenia; jeśli nie zostaną podane żadne argumenty, funkcja zwróci tablicę z aktualnymi ustawieniami.

Wszystkie wspomniane parametry można ustawiać w pliku *php.ini*, jednak często lepiej mieć możliwość zmieniania opcji sesji w fazie wykonania, zwłaszcza w przypadku czegoś tak charakterystycznego jak *cookies*.

Funkcja `session_get_cookie_params()` zwraca tablicę zawierającą ustawienia czasu życia, ścieżki i domeny cookie, a więc można zastosować `session_get_cookie_params()` do upewnienia się, że wykonanie `session_set_cookie_params()` zakończyło się powodzeniem:

```
<?php
$lifetime = time() + 8600;
$path = '/cookiepath';
$domain = 'php.net';
if (session_set_cookie_params ($lifetime, $path, $domain)) {
    $cookie_params = session_get_cookie_params();
    if ($cookie_params[lifetime] == $lifetime &&
        $cookie_params[path] == $path &&
        $cookie_params[domain] == $domain) {
            print "Gratulacje, funkcja session_set_cookie_params() naprawdę
            ↳zadziałała";
    }
}
?>
```

10.5. Wyrejestrowywanie zmiennej z sesji

Musisz usunąć zmienną sesji lub wyrejestrować ją przed zakończeniem wykonywania skryptu.

Technika

Aby usunąć zmienną z sesji, zamiast `unset()` zastosuj funkcję `session_unregister()`:

```
<?php
session_register('somevar');
if (session_is_registered('somevar')) {
    session_unregister('somevar') or die('Nie można wyrejestrować somevar');
}
?>
```

Uwagi

Funkcja `session_unregister()` usuwa wskazaną zmienną z rejestru sesji, a więc w momencie zapisywania sesji nie będzie ona już zawierać wyrejestrowanej zmiennej. Funkcja ta nie usuwa jednak zawartości zmiennej.

10.6. Usuwanie wszystkich zmiennych sesji

Chcesz usunąć wszystkie zmienne z bieżącej sesji.

Technika

Aby wymazać sesję, zastosuj funkcję `session_destroy()`:

Skrypt 10.4.

```
<?php

session_start();

session_register("foo");
session_register("foobar");
session_register("foobarina");

$foo = array ("banan", "jabłko", "pomarańcza", "mango");
$foobar = "owoce";
$foobarina = "warzywa";

session_destroy();

print $foobar;
?>
```

Uwagi

Funkcja `session_destroy()` usunie wszystkie dane sesji ze schowka sesji. Zauważ jednak, że nie zostaną usunięte zawartości zmiennych sesji. Aby to osiągnąć, będziesz musiał zastosować funkcję `session_unset()` i przejść przez wszystkie zmienne sesji, usuwając je z tabeli symboli.

10.7. Stosowanie obiektu jako zmiennej sesji

Masz problemy z ładowaniem obiektów z sesji.

Technika

Pamiętaj, że w każdym miejscu, w którym używasz obiekt, musisz dołączyć definicję klasy.

obiekty/std_class.inc

```
<?php
//
// Plik: std_class.inc
//   Zawiera definicję klasy niezbędną do tego,
//   by obiekt mógł być zmienną sesji.
//
class Foo
{
    var $name;
    var $email;
```

```
    //
    // Prosta funkcja ukazująca istotę sprawy
    //
    function normalize_name()
    {
        $name = preg_replace("/h(.)+/i", "\\1", $this->name);
        return substr($name, 0, 15);
    }
}
?>
```

obiekty/main.php

```
<?php
//
// Plik: main.php
//   Tu będziemy zapisywać i odczytywać obiekt
//
include_once 'std_class.inc';

session_register('foobar');

if (!$foobar) {
    $foobar = new Foo;
    $foobar->name = "Sterling Hughes";
    $foobar->email = "sterling@php.net";
    $foobar->normalize_name();
}
?>
<a href="nextPage.php">Kliknij tutaj</a>
```

obiekty/nextPage.php

```
<?php
//
// Plik: nextPage.php
//   Wyświetla imię nie inicjując klasy ani nie ustawiając zmiennych
//
include_once 'std_class.inc';

session_register('foobar');
print $foobar->name;
?>
```

Uwagi

W czasie, gdy powstawała ta książka, aby zapewnić prawidłowe funkcjonowanie modułu sesji PHP z obiektami, konieczne było dołączenie definicji klasy przed rozpoczęciem sesji. Operacja taka musi zostać wykonana w przypadku każdego obiektu przechowywanego w sesji. W następnych wersjach PHP może się to zmienić, jednak obecnie czynność taka jest niezbędna. Jeśli definicja klasy nie zostanie dołączona, każdorazowa próba uzyskania dostępu do metody lub właściwości obiektu spowoduje wyświetlenie przez PHP ostrzeżenia informującego o konieczności wcześniejszego dołączenia definicji klasy.

10.8. Kodowanie danych

Chcesz zapisać wszystkie zmienne sesji w formie łańcucha znaków i z powrotem wczytać je do zmiennych w późniejszym czasie.

Technika

Zastosuj funkcje `session_encode()` i `session_decode()`:

kodowanie/write.php

```
<?php
session_register("monkey");
$monkey = array("widzieć", "robić");

$string_monkey = session_encode();
$fp = @fopen ("save_monkey.txt", "w") or die ("Nie można otworzyć
↳save_monkey.txt");
@fwrite ($fp, $string_monkey);
@fclose ($fp) or die ("Nie można zamknąć save_monney.txt");
print "Czynności zapisane";
?>
```

kodowanie/read.php

```
<?php
$fp = @fopen("save_monkey.txt", "r") or die("Nie można otworzyć save_monkey.txt");
$data = fread($fp, filesize("save_monkey.txt"));
@fclose ($fp) or die("Nie można zamknąć save_monkey.txt");

session_decode($data);
foreach ($monkey as $action) {
    print "$action\n<br>\n";
}
?>
```

Opis

Funkcja `session_encode()` pobiera wszystkie informacje bieżącej sesji i koduje je do postaci łańcucha znaków, który może być parsowany przez funkcję `session_decode()`. Następnie `session_decode()` parsuje ten łańcuch znaków i na podstawie zawartych w nim informacji tworzy zmienne sesji.

10.9. Tworzenie koszyka na zakupy z zastosowaniem sesji w PHP

Następna część tego rozdziału będzie dotyczyć przechowywania zmiennych i danych, myślę jednak, że zanim do niej przejdziemy, część rozdziału dotycząca sesji powinna zostać zakończona skryptem. Napisałem więc niewielki skrypt koszyka na zakupy

zawierający wszystkie niezbędne funkcje (dodawanie produktów, usuwanie ich z koszyka i podglądanie). Zwróć jednak uwagę, że w skrypcie tym brakuje fragmentu realizującego wysłanie zamówienia — to wykracza już poza temat naszych rozważań.

koszyk/site_lib.inc

```
<?php
//
// site_lib.inc -->
// Zawiera funkcję LoadProducts().
//

// Tablica globalna ze wszystkimi produktami
// Lista $master_products_list wypełniana przez funkcję LoadProducts().
$master_products_list = array();

//
// void LoadProducts(void)
//   Ładuje wszystkie produkty, jakie mogą znaleźć się w koszyku,
//   do globalnej tablicy $master_products_list.
//
function LoadProducts() {
    global $master_products_list;
    $filename = 'products.txt';

    $fp = @fopen($filename, "r") or die("Nie można otworzyć $filename");

    // Na wszelki wypadek nakładamy blokadę współużytkowaną;
    // miałaby znaczenie, gdybyśmy również zapisywali do tego pliku.
    @flock($fp, 1) or die("Nie można nałożyć blokady współużytkowanej na
    ↳$filename");

    while ($line = fgets($fp, 1024)) {
        list($id, $name, $desc, $price) = explode('|', $line);
        $id = trim($id); // obetnij niepotrzebne znaki
        $master_products_list[$id] = array("name" => $name,
                                           "desc" => $desc,
                                           "price" => $price);
    }

@fclose($fp) or die("Nie można zamknąć $filename");
}
?>
```

koszyk/products.txt

```
2kd230 | Rower | Najfajniejszy rower na świecie | 23.83
dksk21 | Sony Playstation | Ekscytująca gra video | 123.00
```

koszyk/cart.php

```
<?php
//
// cart.php: plik główny
//
require 'site_lib.inc';

session_register('cart'); // Rejestrujemy naszą sesję

// Inicjuj koszyk, jeśli nie został jeszcze zainicjowany
```

```
if (!isset($cart[num_items])) {
    $cart = array("num_items" => 0, "products" => array());
}

// Załaduj tablicę $master_products_list z site_lib.inc
LoadProducts();
?>

<html>
<head>
    <title>Sklep z zabawkami Strelinga</title>
</head>

<body>

<h1>Witaj w sklepie z zabawkami Sterlinga</h1>

<?php
if ($cart[num_items]) { // Jeśli jest coś do pokazania
?>
<h2>Produkty znajdujące się w Twoim koszyku</h2>
<br>
<table border="2" cellpading="5" cellspacing="2">
<tr>
    <th>
        Nazwa produktu
    </th>
    <th>
        Krótki opis
    </th>
    <th>
        Cena
    </th>
    <th>
        Ilość
    </th>
    <th>
         
    </th>
</tr>
<?php
    // Przejdź przez wszystkie produkty
    foreach ($cart[products] as $i => $product) {
        $product_id = $product[0];
        $quantity = $product[1];

        $total += $quantity * (double)$master_products_list[$product_id][price];
?>
<tr>
    <td>
        <?php echo $master_products_list[$product_id][name]; ?>
    </td>
    <td>
        <?php echo $master_products_list[$product_id][desc]; ?>
    </td>
    <td>
        <?php echo $master_products_list[$product_id][price]; ?>
    </td>
    <td>
```

```
        <form action="change_quant.php" method="GET">
        <input type="hidden" name="id" value="<?php echo $i; ?>">
        <input type="text" size="3" name="quantity" value="<?php echo $quantity;
        ?>">
    </td>
    <td>
        <input type="submit" value="Zmień ilość">
        </form>
    </td>
</tr>
<?php
    }
?>
<tr>
    <td colspan="2">
        <b>Razem: </b>
    </td>
    <td colspan="2">
        <?php echo $total; ?>PLN
    </td>
</tr>
</table>
<br>
<br>
<?php
}
?>

<h2>Zabawki dostępne w sklepie z zabawkami Sterlinga</h2>
<br>
<i>
    Mamy w ofercie następujące zabawki:
</i>
<br>
<table border="2" cellpadding="5" cellspacing="2">
<tr>
    <th>
        Nazwa produktu
    </th>
    <th>
        Opis produktu
    </th>
    <th>
        Cena
    </th>
    <th>
         
    </th>
</tr>
<?php
    // Pokaż wszystkie produkty
    foreach ($master_products_list as $product_id => $item) {
?>
<tr>
    <td>
        <?php echo $item[name]; ?>
    </td>
    <td>
        <?php echo $item[desc]; ?>
```

```
    </td>
    <td>
        <?php echo $item[price]; ?>PLN
    </td>
    <td>
        <a href="add_item.php?id=<?php echo $product_id; ?>">
            Dodaj ten produkt do koszyka
        </a>
    </td>
</tr>
<?php
    }
?>
</table>
</body>
</html>
```

koszyk/add_item.php

```
<?php
//
// add_item.php:
//  Dodaje produkt do koszyka na zakupy
//
require 'site_lib.inc'; // LoadProducts()

LoadProducts(); // Ładuje produkty z $master_products_list

// Utwórz globalną tablicę $curr_product
$curr_product = array();

// Przejdź przez produkty i zatrzymaj się przy tym,
// który nas interesuje

foreach ($master_products_list as $prod_id => $product) {
    if (trim($prod_id) == trim($id)) {
        $curr_product = $product;
    }
}

// Zarejestruj sesję
session_register('cart');

if($ordered) { // Jeśli został wybrany produkt
    array_push($cart[products], array(trim($id), $quantity));
    $cart[num_items] += $quantity;
}
?>

<html>
<head>
    <title>
    <?php if($ordered) { ?>
        Dodano <?php echo $curr_product[name]; ?> do koszyka na zakupy
    <?php } else { ?>
        Dodaj <?php echo $curr_product[name]; ?> do koszyka na zakupy
    <?php } ?>
    </title>
</head>
```

```
<body>
<?php if ($ordered) { ?>
    <h1><?php echo $curr_product[name]; ?>
        został dodany do koszyka na zakupy</h1>

    <a href="cart.php">Wróć</a> i kontynuuj zakupy.
<?php } else { ?>
    <h1>Dodaj <?php echo $curr_product[name]; ?> do koszyka na zakupy</h1>

    <form action="<?php echo $PHP_SELF; ?>" method="GET">
    Nazwa produktu: <?php echo $curr_product[name]; ?>
    <br>
    Opis produktu: <?php echo $curr_product[desc]; ?>
    <br>
    Cena produktu: <?php echo $curr_product[price]; ?>
    <br>
    Ilość produktu: <input type="text" size="7" name="quantity">
    <input type="hidden" name="id" value="<?php echo $id; ?>">
    <input type="hidden" name="ordered" value="1">

    <input type="submit" value="Dodaj do koszyka">
    </form>

<?php } ?>
</body>
</html>
```

koszyk/change_quant.php

```
<?php
//
// change_quant.php:
//   Zmienia ilość produktu w koszyku na zakupy.
//
session_register('cart'); // zarejestruj sesję

// Rzutowanie typu na int, aby upewnić się,
// że pobieramy właściwy element
$i = (int)$id;

// Zapisz dotychczasową ilość produktu w celach obliczeniowych
// oraz aby ją wyświetlić
$old_num = $cart[products][$i][1];

if ($quantity) {
    $cart[products][$i][1] = $quantity; // zmień ilość
} else {
    unset($cart[products][$i]); // niech produkt odejdzie w zapomnienie
}

// Uaktualnij liczbę produktów
$cart[num_items] = ($old_num > $quantity) ?
                   $cart[num_items] - ($old_num-$quantity) :
                   $cart[num_items] + ($quantity-$old_num);
?>

<html>
<head>
    <title>
```

```
        Ilość została zmieniona
    </title>
</head>
<body>
    <h1>Ilość zmieniona z <?php echo $old_num; ?> na <?php echo $quantity; ?></h1>
    <a href="cart.php">Wróć</a> i kupuj dalej.
</body>
</html>
```

10.10. Serializacja

Chcesz zapisać zawartość zmiennej w postaci łańcucha znaków.

Technika

Utwórz funkcje implementujące funkcje `serialize()` i `unserialize()`, a następnie odczytuj i zapisuj dane do pliku:

serializacja/loadsave.inc

```
<?php
//
// Plik: loadsave.inc
//   Biblioteka funkcji zapisujących
//   i odczytujących dane do i z pliku
//

//
// int save (string varname):
//   Zapisuje wartość zmiennej varname do pliku.
//
function save($var) {
    global $$var;
    $data = serialize($$var); // łańcuch reprezentujący $$var
    $filename = "php_serialized_vars/" . $var . ".txt";
    $fp = @fopen($filename, "w") or die("Nie można otworzyć $filename do zapisu");

    fwrite($fp, $data);
    @fclose($fp) or die("Nie można zamknąć $filename");
    return(true);
}

//
// int load (string varname)
//   Ładuje wartość zmiennej varname z pliku.
//
function load($var)
{
    global $$var; // Umieść zapisaną zmienną w zasięgu globalnym
    $filename = "php_serialized_vars/" . $var . ".txt";
    $fp = @fopen($filename, "r") or die("Nie można otworzyć $filename do odczytu");

    $data = fread($fp, filesize($filename));
```

```
    @fclose($fp) or die("Nie można zamknąć $filename");

    $$var = unserialize($data);
    return(true);
}
?>
```

Nowo utworzone funkcje można wywoływać w następujący sposób:

serializacja/save.php

```
<?php
include 'loadsave.inc';
$foo = "cześć";
save('foo');
?>
<a href="load.php">Kliknij tutaj</a>
```

serializacja/load.php

```
<?php
include 'loadsave.inc';
load('foo');
print $foo;
?>
```

Uwagi

Funkcja `serialize()` tworzy łańcuchową reprezentację danych PHP, niezależnie od tego, czy dane te mają postać tablicy, obiektu, zwykłego łańcucha znaków czy liczby. Funkcja `unserialize()` odczyta ten łańcuch znaków i przywróci mu oryginalną formę. Jest to dosyć podobne do działania programów kompresji i dekompresji, takich jak *bzip* czy *gzip*. Zauważ, że w przypadku deserializacji obiektów musisz najpierw załadować definicję klasy obiektu, podobnie jak w przypadku sesji.

10.11. Serializacja WDDX

Chcesz przeprowadzić serializację wielu zmiennych naraz, a nie każdej zmiennej oddzielnie, albo chcesz, by zmienne mogły być współużytkowane z innymi językami lub procesami

Technika

Zastosuj funkcje WDDX, które przeprowadzają serializację zgodną ze standardem WDDX opisanym na stronie *http://www.wddx.org/*.

```
<?php
$ice_cream = array("Miętowo-czekoladowe", "Waniliowe", "Czekoladowe", "Kawowe");
$packet_id = wddx_packet_start("PHP");
wddx_add_vars($packet_id, "ice_cream");
$packet = wddx_packet_end($packet_id);
?>
```

Uwagi

Format WDDX, czyli *Web Distributed Data eXchange*, to „... mechanizm wymiany złożonych struktur danych między środowiskami aplikacji" (*http://www.wddx.org/DTD.htm*). Mówiąc w skrócie, przeprowadzanie serializacji z zastosowaniem funkcji WDDX polega na tym, że funkcje te tworzą „pakiety" zawierające informacje na temat zmiennych.

Pakiety te mogą być przechowywane oraz z powrotem wczytywane do programu, podobnie jak robiliśmy to za pomocą funkcji `serialize()` i `unserialize()` w podrozdziale 10.10. Kolejnym popularnym zastosowaniem WDDX jest serializacja zmiennych w formacie WDDX w celu umożliwienia ich odczytywania i wykorzystywania przez inne programy.

10.12. Deserializacja WDDX

Dysponujesz łańcuchem znaków powstałym po serializacji WDDX i chcesz z powrotem wydobyć z niego dane.

Technika

Zastosuj funkcję `wddx_deserialize()` do odkodowania łańcucha WDDX i wczytania go do zmiennej PHP:

Skrypt 10.5.

```
<?php
$favorite_tv_shows = array("M*A*S*H", "Seinfeld", "Simpsonowie");
$text = wddx_serialize_vars("favorite_tv_shows");
$favorite_tv_shows_again = wddx_deserialize($text);

foreach ($favorite_tv_shows_again as $favorite_shows) {
    foreach ($favorite_shows as $show) {
        print "$show\n<br>\n";
    }
}

?>
```

Uwagi

Funkcja `wddx_deserialize()` pobiera pakiet WDDX i konwertuje go do postaci tablicy z odpowiadającymi zmiennymi PHP. Jednym z ciekawszych zastosowań `wddx_deserialize()` jest pobranie pakietów WDDX wygenerowanych przez inne języki i przetłumaczenie ich na zmienne PHP.