helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

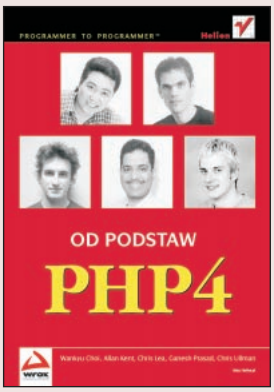
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

PHP4. Od podstaw



Autorzy: Wankyu Choi, Allan Kent, Chris Lea, Ganesh Prasad, Chris Ullman
Tłumaczenie: Paweł Gonera, Piotr Rajca
ISBN: 83-7197-494-9
Tytuł oryginału: Beginning PHP4
Format: B5, stron: około 780

PHP4 jest najnowszym wcieleniem PHP – PHP Hypertext Preprocesor. Jest to język programowania przeznaczony do tworzenia dynamicznych, interaktywnych witryn WWW, który został wymyślony przez Rasmusa Lerdorfa przed rokiem 1994. Od tego czasu zmienił się niezmiernie i jest teraz używany przez programistów WWW na całym świecie. Czym więc tak naprawdę jest? Można go określić technicznie jako przenośny, wbudowywany w HTML język programowania skryptów serwera WWW. PHP4 może być wykorzystany w bardzo różnych rodzajach aplikacji, od użytkowych, jak edytor, do wydajnych witryn typu sklep lub wyszukiwarka. W tek książce spróbujemy przekazać czytelnikowi wiedzę na temat tworzenia witryn przy użyciu PHP4. Przedstawione zostaną użyteczne techniki kodowania i pomysły w jaki sposób można ich użyć we własnych aplikacjach. PHP może być użyty do stworzenia witryn, które zna każdy użytkownik Sieci. Od witryn handlu elektronicznego do wyszukiwarek czy portali informacyjnych. Wiele dużych witryn WWW posiada niektóre lub wszystkie te mechanizmy. W tej książce stworzymy między innymi:

- edytor oparty na przeglądarce pozwalający na tworzenie i edycję plików na serwerze przy użyciu przeglądarki,
- witrynę dla sklepów pozwalającą na odszukanie towaru i wyświetlenie graficznej mapy odpowiedniego stoiska,
- sieciową grę,
- wyszukiwarkę WWW pozwalającą na wyszukanie witryny w katalogu lub nawigację przez strukturę hierarchiczną,
- usługę list pocztowych pozwalającą użytkownikom na prenumeratę listy i pozwalającą administratorom na wysyłanie poczty do subskrybentów.

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

Spis treści

O Autorach 11

Wstęp 13

Dlaczego PHP4? 14
Znak zachęty 15
Czego potrzebujesz, aby korzystać z książki? 15
Zasoby w sieci dotyczące PHP4 16
Konwencje 16

Rozdział 1. Instalacja 19

Co powinienem zrobić? 20
Instalowanie PHP4 na Windows 21
Instalowanie PHP do Apache Web Server 23
Instalowanie PHP4 na Linuksie i innych systemach uniksowych 24
Wybór metody instalacji 25
Skąd otrzymać pliki RPM? 25
Jakich pakietów potrzebujemy? 26
Konfigurowanie i uruchamianie Apache i PHP4 27
Testowanie instalacji 30

Rozdział 2. Pisanie programów w PHP 35

Przykładowy program PHP 36
Przeglądanie strony WWW 40
Klient-serwer 40
Na scenę wchodzi PHP 46
Skrypty serwera 47
Pamięć podręczna 48
Zmienne 49
Typy danych 52
Stałe 60
Inicjalizacja 61
Konwersje 62
Zmienne środowiska 65
Podsumowanie 66

Rozdział 3. Pobieranie danych od użytkownika 67

Formularze HTML 68
Znacznik FORM 68
Atrybuty FORM 69
Kontrolki formularzy HTML a PHP 73
Użycie wartości z formularza w skryptach PHP 96
Podsumowanie 102

Rozdział 4. Podejmowanie decyzji 103

Wyrażenia warunkowe lub rozgałęzianie 104
Przykład rozgałęziania z życia wzięty 104
Instrukcja if 105
Wielokrotne warunki — else i elseif 121
Instrukcja switch 130
Kontrola poprawności formularza 134
Podsumowanie 139

Rozdział 5. Pętle i tablice 141

Pętle 142
Pętle while 142
Pętle do while 149
Pętle for 153
Tablice 158
Inicjalizacja tablic 159
Przeglądanie tablic 161
Sortowanie tablic 170
Różne funkcje operujące na tablicach 173
Tablice wielowymiarowe 176
Praktyczne zastosowanie tablic 177
Nowe własności pętli i tabel i PHP4 181
Wielokrotne sortowanie tablic 181
Pętle foreach 182
Podsumowanie 184

Rozdział 6. Organizacja kodu 187

Dlaczego ponowne użycie kodu jest tak przydatne? 188
Modularyzacja 188
Funkcje 189
Definiowanie i wywoływanie funkcji 189
Wybór funkcji 196
Przekazywanie wartości 197
Zasięg zmiennych 201
Zmienne globalne i lokalne 202
Zagłębianie 208
Rekurencja 210
Pliki dołączane 212
Typowe zastosowania plików dołączanych 216
Podsumowanie 218

Rozdział 7. Obsługa i unikanie błędów 219

Obsługa błędów w PHP 220
Niebezpieczne komunikaty 220
Nieczytelne strony WWW 220
Niewidoczne komunikaty błędów 221
Typy błędów 221
Błędy składni 221
Błędy logiczne 224

Zasady dobrego kodowania ... 229
Wcinanie kodu ... 229
Komentowanie kodu ... 230
Użycie funkcji ... 231
Użycie dołączanych plików ... 232
Użycie znaczących nazw zmiennych ... 232
Próby załamania kodu ... 233
Dokładne sprawdzanie formularza ... 234
Pobieranie danych od użytkownika ... 236
Wyrażenia regularne ... 236
Wzorce ... 237
Znaki specjalne ... 238
Uruchamianie skryptów PHP ... 249
Użycie echo() ... 249
Sprawdzanie źródła HTML ... 250
Ukrywanie komunikatów błędów ... 251
Sprawdzanie dziennika błędów ... 251
Ręczne wykonywanie ciężkiej pracy ... 252
Podsumowanie ... 252

Rozdział 8. Praca z klientem ... 255

Ulepszanie protokołu bezstanowego ... 256
Rozmowa z użytkownikiem — HTTP, HTML, PHP i interakcyjność ... 257
Sesje w PHP4 ... 258
Sposoby typu „Zrób to sam” ... 259
Użycie ukrytych pól ... 259
Bardziej uniwersalna metoda ... 263
Ciągi zapytań ... 267
Cookie ... 275
Sesje ... 283
Sesje PHP4 ... 283
Podsumowanie ... 287

Rozdział 9. Obiekty ... 289

Terminologia obiektowa ... 290
Użycie predefiniowanych klas ... 292
Tworzenie kalkulatora z pamięcią ... 298
Dalsze poznawanie naszej prostej klasy ... 301
Tworzenie własnych klas ... 304
Tworzenie klasy od podstaw ... 304
Rozszerzanie istniejącej klasy ... 307
Użyteczny obiekt ... 310
Podsumowanie ... 314

Rozdział 10. Obsługa plików i katalogów ... 315

Działania na plikach ... 316
Otwieranie i zamykanie plików ... 316
Czytanie i zapis do pliku ... 319
Pobieranie informacji o plikach ... 331
Rozdzielanie nazwy pliku i ścieżki ... 339

Kopiowanie, zmiana nazwy i usuwanie plików ... 339
Tworzenie edytora tekstu ... 342
Praca z katalogami ... 350
Inne funkcje operujące na katalogach ... 351
Przeglądanie drzewa katalogów ... 353
Tworzenie przeglądarki katalogów ... 354
Przesyłanie plików ... 359
Łączymy wszystko razem — sieciowy edytor tekstu ... 362
Zasoby ... 368
Podsumowanie ... 368

Rozdział 11. Korzystanie z baz danych w PHP ... 369

Bazy danych ... 369
Modele danych ... 370
Architektura baz danych ... 373
Dlaczego MySQL? ... 375
Instalacja MySQL-a ... 376
Prezentacja SQL ... 380
Wykorzystanie MySQL-a ... 387
Uruchamianie programu klienta ... 387
Wybór używanej bazy danych ... 388
Poznawanie danych w bazie ... 388
Operowanie na danych przechowywanych w bazie ... 390
Użycie poleceń GRANT i REVOKE ... 392
Podsumowanie ... 394
Obsługa MySQL w PHP ... 394
Podstawowe funkcje ... 394
Tworzenie baz danych i tabel w MySQL ... 402
Tworzenie przykładowej bazy danych oraz tabel w PHP ... 407
Modyfikacja tabel ... 410
Wstawianie danych do tabel ... 412
Zasoby ... 416
Podsumowanie ... 416

Rozdział 12. Pobieranie danych z baz MySQL w skryptach PHP ... 419

Pobieranie danych przy użyciu PHP ... 419
Polecenia SQL służące do pobierania danych ... 423
Funkcje serwera ... 423
Pobieranie wartości wybranych pól ... 424
Tworzenie podsumowań ... 429
Bardziej złożone zapytania ... 430
Zastosowanie wszystkich poznanych informacji ... 434
Użycie przeglądarki rekordów ... 443
Zasoby ... 444
Podsumowanie ... 444

Rozdział 13. Operowanie na danych w bazach MySQL w skryptach PHP ... 445

Wstawianie rekordów w skryptach PHP ... 445
Aktualizacja i usuwanie rekordów z tabel ... 448
Operowanie na polach dat i czasu ... 451
Uzyskiwanie informacji o tabelach bazy danych ... 455

Tworzenie skryptu rejestrującego użytkowników ... 464
Tworzenie skryptu rejestrującego wizyty ... 471
Tworzenie skryptu zarządzającego użytkownikami ... 480
Zasoby ... 489
Podsumowanie ... 490

Rozdział 14. XML ... 491

Czym jest XML? ... 491
Struktura dokumentu XML ... 493
Poprawnie sformułowane dokumenty XML ... 494
DTD ... 498
Przetwarzanie sterowane zdarzeniami ... 500
Analiza przykładowego pliku XML ... 501
Przetwarzanie pliku zewnętrznego ... 508
Podsumowanie ... 513

Rozdział 15. Obsługa poczty elektronicznej ... 515

Wysyłanie wiadomości poczty elektronicznej w PHP ... 515
Anatomia wiadomości poczty elektronicznej ... 518
Obsługa załączników ... 526
Anatomia wiadomości poczty elektronicznej raz jeszcze ... 527
Dołączanie plików do wiadomości poczty elektronicznej ... 531
Skrypt obsługujący biuletyn informacyjny ... 539
Zasoby ... 556
Podsumowanie ... 556

Rozdział 16. Generacja grafiki ... 559

Wiadomości podstawowe ... 559
Tworzenie obrazu ... 561
Rysowanie ... 563
Wykorzystanie wszystkich poznanych możliwości ... 567
Zastosowanie praktyczne ... 570
Interaktywne mapy ... 571
Rozpoczynamy pracę ... 571
Tworzenie szkieletu aplikacji ... 578
Dalsze możliwości funkcjonalne ... 584
Wyświetlanie szczegółowych informacji o sklepie ... 586
Zaawansowane techniki obsługi obrazów ... 590
Stylizowane mapy ... 590
Ograniczenia palet ... 594
Podsumowanie ... 596

Rozdział 17. Studium zagadnienia: katalog adresów URL ... 597

Narzędzie do zarządzania katalogiem adresów URL ... 598
Projekt katalogu adresów URL ... 598
Wymagania użytkowników ... 599
Interfejs użytkownika ... 601
Przechowywanie informacji ... 602
Inne ograniczenia mające wpływ na projekt ... 607
Struktura kodu ... 607

Implementacja kodu ... 611
Kod wspólny — php_directory.inc ... 611
Skrypt użytkownika — php_directory.php ... 656
Skrypt administracyjny — dir_manager.php ... 657
Informacje od użytkowników ... 680
Podsumowanie ... 680

Dodatek A ODBC ... 681

Czym jest ODBC? ... 681
Czym ODBC nie jest? ... 683
PHP i ODBC ... 686
Nawiązywanie połączenia ze źródłem danych ... 687
Wykonywanie poleceń SQL ... 687
Obsługa wyników ... 688

Dodatek B Funkcje PHP ... 695

Funkcje analizatora składni XML ... 695
Funkcje Aspell ... 697
Funkcje CURL (Client URL Library) ... 697
Funkcje FDF (Forms Data Format) ... 697
Funkcje graficzne ... 699
Funkcje HTTP ... 702
Funkcje kontroli generacji wyników ... 702
Funkcje matematyczne ... 703
Funkcje obsługi baz danych ... 705
Funkcje obsługi baz danych dBase ... 705
Funkcje obsługi baz danych DBM ... 706
Funkcje obsługi baz danych Informix ... 707
Funkcje obsługi baz danych InterBase ... 709
Funkcje obsługi baz danych Microsoft SQL Server ... 710
Funkcje obsługi baz danych mSQL ... 711
Funkcje obsługi baz danych MySQL ... 713
Funkcje obsługi baz danych Oracle 8 ... 716
Funkcje obsługi baz danych Oracle ... 717
Funkcje obsługi baz danych PostgreSQL ... 718
Funkcje obsługi baz danych Sybase ... 721
Funkcje warstwy abstrakcji bazy danych ... 722
Funkcje Zunifikowanego ODBC ... 723
Funkcje obsługi błędów i rejestracji ... 725
Funkcje obsługi dat i czasu ... 726
Funkcje obsługi DOM XML ... 727
Funkcje obsługi funkcji ... 727
Funkcje obsługi kalendarzy ... 727
Funkcje obsługi katalogów ... 729
Funkcje obsługi klas i obiektów ... 729
Funkcje obsługi łańcuchów znaków ... 730
Funkcje obsługi opcji PHP oraz informacyjne ... 734
Funkcje obsługi poczty elektronicznej ... 736
Funkcje obsługi sesji ... 736
Funkcje obsługi systemu plików ... 737
Funkcje obsługi tablic ... 740

Funkcje obsługi wyrażeń regularnych (rozszerzony standard POSIX) 744
Funkcje obsługi wyrażeń regularnych Perla 745
Funkcje obsługi zmiennych 745
Funkcje POSIX 747
Funkcje Pspell 748
Funkcje serwera Apache 749
Funkcje sieciowe 750
Funkcje sterowania wykonywaniem programów 751
Funkcje URL 752
Funkcje WDDX 752
Funkcje Zlib 753
Inne funkcje 754

2

Pisanie programów w PHP

W pierwszym rozdziale przedstawiliśmy PHP, opisaliśmy proces instalacji i konfiguracji serwera WWW oraz PHP. Do tej pory nie pokazaliśmy szczegółowo żadnego przykładu kodu PHP poza tym, który posłużył do przetestowania poprawności instalacji. Pierwszym zagadnieniem, jakim zajmiemy się w tym rozdziale, będzie napisanie bardzo prostej strony PHP i uruchomienie jej na serwerze WWW. W procesie tworzenia tej strony opowiemy, w jaki sposób działa i co robi przykładowy kod. Zapoznamy się również z zadaniami serwera WWW oraz dokładnie opiszemy działanie PHP.

Po poznaniu operacji, jakie są wykonywane przez PHP, zajmiemy się podstawowymi elementami języka oraz sposobami przechowywania informacji używanych na stronach WWW. Każdy język programowania posiada mechanizm zapamiętywania danych i kojarzenia ich z identyfikatorem, używanym do odwołania się do danych w dalszej części programu. W jaki sposób przechowujesz informacje o datach, na przykład o urodzinach Twoich znajomych? Jak korzystasz z tej listy i w skąd wiesz, która data jest związana z czyimi urodzinami? PHP, tak samo jak większość języków programowania, używa do tego celu mechanizmu zmiennych. Ostatnia część tego rozdziału jest poświęcona korzystaniu ze zmiennych, operacjom matematycznym oraz prostym manipulacjom na tekstach.

W rozdziale tym będziemy zajmować się następującymi zagadnieniami:

- napisaniem i przeanalizowaniem bardzo krótkiego programu w PHP,
- przedyskutowaniem niektórych protokołów internetowych,
- przedstawieniem ról maszyny PHP i serwera WWW,
- zdefiniowaniem znaczenia interpretacji i wykonania,
- opisaniem zmiennych — czym są?
- typami danych — omówieniem typów danych, jakie mogą przyjmować zmienne,
- operacjami możliwymi na zmiennych,
- stałymi,
- konwersją zmiennych z jednego typu danych na inny,

- zmiennymi środowiska.

Przykładowy program PHP

Rozdział rozpoczniemy nieomal najprostszym możliwym jednolinijkowym przykładem, w którym chcemy pokazać, że strony PHP składają się z trzech składników: tekstu, kodu HTML i skryptów PHP. Strony zawierające skrypty PHP są inne niż te, które zawierają jedynie HTML i w celu poinformowania o tym analizatora PHP są zapisane na serwerze WWW w plikach z rozszerzeniem *.php* (lub podobnym), na tej podstawie są rozpoznawane i wykonywane przez PHP zainstalowany na serwerze.

> Rozszerzenie *.php* jest zależne od określonej konfiguracji PHP. Jeżeli chcesz, możesz tak skonfigurować PHP, aby korzystał z nieomal dowolnego rozszerzenia, na przykład *.dlugierozszerzenie*. W tej książce korzystamy jedynie z rozszerzenia *.php* pisanego małymi literami.

Wyniki działania skryptu są przesyłane do przeglądarki w postaci HTML. Później zajmiemy się tym nieco dokładniej, teraz przedstawimy przykład.

Pierwszy przykładowy program

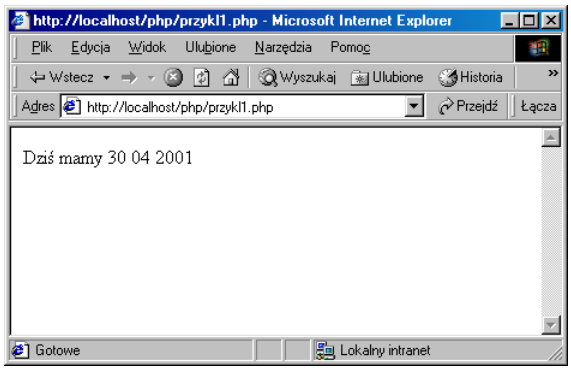
1. Uruchom Twój ulubiony edytor stron WWW i wpisz następujący tekst:

```
<HTML>
<BODY>
Dziś mamy
<?php echo gmdate("d m Y");
?>
</BODY>
</HTML>
```

2. Zapisz tekst w pliku *przykl1.php* w katalogu głównym dokumentów serwera WWW.

3. Otwórz Twoją ulubioną przeglądarkę i wpisz pełny adres URL serwera i strony WWW, dla tego przykładu jest to *http://localhost/przykl1.php*. Powinieneś uzyskać wynik podobny do poniższego w programie Internet Explorer.

Rysunek 2.1.



Różne rodzaje kodu

Tak jak zapowiedzieliśmy wcześniej, najpierw skupimy się na przedstawieniu trzech różnych typów kodu użytego na stronie. Spójrzmy na wpisany właśnie kod i sklasyfikujmy każdy z tych trzech typów:

```
<HTML>
<BODY>
```

```
Dziś mamy
```

```
<?php echo gmdate("d m Y");
?>
</BODY>
</HTML>
```

Kod bez żadnego wyróżnienia właściwie nie jest kodem. Jest to po prostu **tekst**. Nic więcej na jego temat nie można powiedzieć.

Linie w ramce są **znacznikami HTML**. Znaczniki dowolnego typu są łatwe do rozpoznania, ponieważ są oznaczane przez otwierające i zamykające **nawiasy trójkątne**. Nie jest to prawdziwy kod programu w tradycyjnym rozumieniu tego określenia, dlatego do opisania kodu HTML używamy słowa znacznik. Zakładamy, że znasz HTML, więc nie będziemy więcej mówić na ten temat.

Dodamy jeszcze, że wszystkie znaczniki HTML w tej książce będziemy pisać wielkimi literami, mimo że jest to niezgodne ze standardem (standard HTML 4.0 określa, że wszystkie znaczniki powinny być pisane małymi literami). Robimy tak, ponieważ nie chcemy używać ciągle niewygodnego zaznaczania w całej książce, a wszystkie przeglądarki i tak nie rozróżniają wielkości liter w znacznikach.

> Wszyscy, którzy znają już XHTML powinni wiedzieć, że język ten, zgodny ze standardem HTML 4.01 (najnowszy standard w XML), jest nietolerancyjny w wielu aspektach, które przeglądarki HTML najczęściej ignorowały. Oznacza to, że XHTML nie pozwoli na używanie znaczników, które wyszły z użycia lub na nieprawidłowe zamykanie znaczników. Wymaga również pisania kodu HTML małymi literami. Do tej pory nie ma popularnych przeglądarek akceptujących jedynie XHTML, więc nie powinniśmy obawiać się negatywnych skutków zastosowania wielkich liter.

Kod zaznaczony ciemnym tłem jest **skryptem PHP**. Skrypt PHP jest również zaznaczony nawiasem trójkątnym i znakiem zapytania. Za każdym razem, gdy zobaczysz fragment tekstu ograniczony znacznikami <?php i ?>, powinieneś wiedzieć, że cały tekst w środku musi być skryptem PHP.

Jak widać na wynikowej stronie HTML, te trzy typy tekstu harmonijnie ze sobą współpracują, pomimo to skrypt PHP musi być przetworzony przez serwer WWW (który może być na innym komputerze niż twoja przeglądarka).

Jak działa ten kod?

Rozpoznając naturę różnych typów kodu na stronach WWW, nie wyjaśniliśmy, co robi każda z linii kodu PHP. Naprawimy to teraz, na naszej stronie zawarta była jedna linia skryptu PHP:

```
echo gmdate("d m Y");
```

W linii tej wykonywane są trzy czynności. Na początku mamy słowo `echo()`, które jest instrukcją PHP, pobierającą cokolwiek do niej zostanie przekazane i drukującą to na stronie WWW. Jeżeli do skryptu wpiszemy linię:

```
echo "Witaj świecie";
```

na stronie będącej wynikiem działania tego skryptu powinieneś zobaczyć napis `"Witaj świecie"`. Możesz zauważyć, że w naszym przykładzie na ekranie pojawia się data, a do funkcji `echo()` przekazaliśmy jedynie **funkcję** `gmdate("d m Y")` — jest to druga wykonywana czynność, którą opiszemy za chwilę.

> PHP posiada ogromną bibliotekę zarezerwowanych słów zwanych funkcjami, które służą do realizacji wielu częstych zadań, takich jak zwracanie daty i czasu, wysyłanie poczty, wykonywanie wielu skomplikowanych operacji matematycznych lub wstrzymywanie wykonania skryptu na kilka sekund. W dodatku B przedstawiliśmy całą listę funkcji. Nie musisz znać ich wszystkich na pamięć, ale wielu z nich użyjemy w przykładach w tej książce, a później będziesz z nich stale korzystał.

Ostatnim elementem jest średnik. PHP wymaga, aby każda linia kodu (jak się przekonamy, istnieją nieliczne wyjątki od tej reguły) była zakończona średnikiem w celu oznaczenia jej końca.

Zauważ, że słowo `gmdate()` nie było otoczone cudzysłowami. W naszym przykładzie `"Witaj świecie"`, otaczając tekst do wyświetlenia cudzysłowami, wskazaliśmy PHP, że chcemy wyświetlić go dokładnie w taki sposób, jak został napisany. W naszym przypadku, niepodając cudzysłowów, nakazujemy PHP użycie funkcji `gmdate()` i zwrócenie jej wyniku. Funkcja zwraca bieżący czas w postaci czasu Greenwich. Przedyskutujemy teraz dokładniej, jak działa funkcja `gmdate()`.

Pewnie zauważyłeś, że w naszym przykładzie wyświetlamy datę w postaci:

```
30 04 2001
```

Jest jednak wiele innych formatów wyświetlania daty. Mogłeś, na przykład otrzymać wynik w postaci:

```
30/4/00 09-30AM
```

lub

```
Monday, 30 April
```

Są to prawidłowe formaty dat, jednak PHP nie odróżnia, którego formatu chcesz używać lub czy przypadkiem nie chcesz zastosować całkowicie innej wersji. Musisz więc wskazać, co ma zrobić i jest to realizowane przez sekcję zapisaną w nawiasach i cudzysłowach po poleceniu `gmdate`. W naszym przykładzie podaliśmy, że na początku daty ma być dzień, następnie miesiąc i rok. Wielkość liter D, M i Y użytych w funkcji jest również znacząca. PHP interpretuje litery przedstawione w tabeli 2.1. w różny sposób.

Tabela 2.1.

Znak	Wynik
a	Wyświetla am lub pm.
A	Wyświetla AM lub PM.
d	Dwucyfrowy dzień miesiąca z poprzedzającym zerem, tj. 01 do 31.
D	Tekstowo dzień tygodnia w postaci 3 liter, na przykład: Fri.
F	Miesiąc tekstowo, w pełnej postaci, na przykład: January.
h	Godzina w formacie 12-godzinnym tj. od 01 do 12.
H	Godzina w formacie 24-godzinnym tj. od 00 do 23.
g	Godzina w formacie 12-godzinnym bez początkowych zer tj. od 1 do 12.
G	Godzina w formacie 24-godzinnym bez początkowych zer tj. od 0 do 23.
i	Wyświetla minuty od 00 do 59.
j	Dzień miesiąca bez początkowych zer od 1 do 31.
l	Dzień tygodnia w pełnej postaci, na przykład Friday.
L	Wskazuje, czy jest to rok przestępny jako 0 lub 1.
m	Wyświetla miesiąc 01 do 12.
n	Wyświetla miesiąc bez początkowych zer, tj.: od 1 do 12.
M	Miesiąc w skróconej, trzyliterowej postaci, na przykład Jan.
s	Wyświetla sekundy, od 00 do 59.
S	Przyrostek dla liczebników w języku angielskim jako dwa znaki, na przykład th, nd.
t	Ilość dni w podanym miesiącu, tj. od 28 do 31.
T	Ustawienie strefy czasowej na serwerze, na przykład MDT.

U	Ilość sekund od początku epoki.
w	Pokazuje dzień tygodnia w postaci numerycznej, od 0 (niedziela) do 6 (sobota).
Y	Podaje rok w postaci czterocyfrowej, na przykład: 1999.
y	Podaje rok w postaci dwucyfrowej, na przykład: 99.
z	Zwraca dzień w roku, od 0 do 365.
Z	Podaje przesunięcie strefy czasowej w sekundach (od –43200 do 43200) .

Jest to dosyć obszerna lista. Nie będziemy umieszczać takiej tabeli w tekście za każdym razem, gdy przedstawiamy nową funkcję. Czasami będziesz musiał odszukać potrzebne Ci informacje w dodatkach. Możemy również zmienić nasz poprzedni przykład na:

```
echo gmDate("D");
```

Zostanie wtedy wyświetlone:

```
Tue
```

Jako zadanie domowe pozostawiamy sprawdzenie, jakie dane możemy wyświetlić, modyfikując ten przykład. Teraz, gdy lepiej rozumiemy, jak działa przedstawiony kod, możemy zająć się następnymi zagadnieniami.

Przeglądanie strony WWW

Popatrzmy, co dzieje się z naszą stroną pomiędzy jej napisaniem i wyświetleniem. Gdy w rozdziale 1. instalowaliśmy PHP, proces ten został podzielony na części, ponieważ instalowaliśmy różne elementy oprogramowania.

Jednym z tych elementów jest serwer WWW, który udostępnia nasze strony wszystkim. Innym zadaniem serwera jest zarządzanie obszarem (zwykle katalogiem lub strukturą katalogów), w którym umieszczamy nasze pojedyncze strony lub całą witrynę WWW. Istnieje wiele bezpłatnych i komercyjnych serwerów, ale dominują serwery dwóch firm, zajmujące ponad 70% rynku. Te dwa produkty to serwer Apache oraz Microsoft Internet Information Server. Zakładamy, że do udostępnienia swoich stron używasz jednego z nich.

Gdy korzystasz z sieci, oglądając strony WWW, automatycznie łączysz się z serwerem. Proces wysyłania adresu URL jest określany **żądaniem**. Serwer interpretuje adres URL, odszukuje odpowiednią stronę i odsyła ją do przeglądarki. Jest to nazywane **odpowiedzią**. Przeglądarka zapamiętuje otrzymany kod i tworzy z niego obraz strony. Przeglądarka jest w tym dialogu klientem, natomiast cały proces jest zgodny z modelem **klient-serwer**.

Klient-serwer

Termin ten dobrze opisuje sposób działania, określając zadania dystrybucji. Serwer (serwer WWW) przechowuje, interpretuje i rozsyła dane, natomiast klient (przeglądarka) łączy się z serwerem w celu uzyskania danych. Od tej pory, jeżeli użyjemy określenia klient, będziemy mieli na myśli przeglądarkę. Jak się okaże w późniejszych rozdziałach, termin klient-serwer jest uproszczeniem procesu — to nieco abstrakcyjny opis rzeczywistości. Aby zrozumieć szczegóły, musimy krótko opisać sposób działania samego Internetu.

Protokoły internetowe

Nie chcemy opisywać tu całej historii Internetu, najważniejsze, że jest to sieć połączonych ze sobą komputerów. Internet został zaprojektowany do przenoszenia *danych* z jednego miejsca do drugiego. Do tego celu używa zestawu protokołów sieciowych (znanych jako **TCP/IP**).

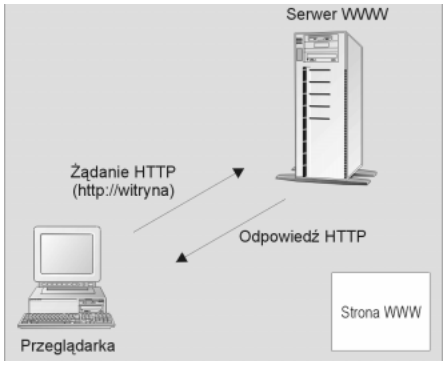
> **Protokół sieciowy** jest metodą opisu pakietów danych przesyłanych pomiędzy węzłami przez sieć telefoniczną, kablową bądź linię T1.

Jedną z zalet protokołu TCP/IP jest to, że można szybko zmienić trasę przesyłania danych w przypadku awarii lub spowolnienia jednego z segmentów sieci. Gdy użytkownik zamierza pobrać za pomocą przeglądarki stronę WWW, przeglądarka realizuje to polecenie przy użyciu protokołu o nazwie **Transmission Control Protocol** (TCP). TCP jest protokołem transportowym, zapewniającym niezawodny format transmisji dla tego typu żądań. Zapewnia, że cały komunikat zostanie poprawnie przekształcony na postać odpowiednią do transmisji (oczywiście prawidłowo rozpakowany i połączony po dotarciu do miejsca przeznaczenia). Protokół sieciowy TCP/IP jest metodą opisu pakietów danych, które są przesyłane między węzłami za pomocą linii telefonicznej, kablowej lub linii *T1*.

Zanim dane zostaną podzielone w celu wysłania w sieć, muszą otrzymać odpowiedni adres. Drugi z protokołów sieciowych, **HyperText Transfer Protocol** (**HTTP**), nakleja na pakiety etykietki adresowe, aby TCP/IP odpowiednio kierował dane. HTTP jest protokołem używanym przez sieć WWW do przesyłania danych między komputerami. Jeżeli zobaczysz adres URL rozpoczynający się od *http://*, powinieneś wiedzieć, że korzysta on z protokołu HTTP. Można również porównać TCP/IP do poczty, która zajmuje się przesyłaniem przesyłek, natomiast HTTP do znaczka i adresu na kopercie, które zapewniają, że dotrze ona do miejsca przeznaczenia.

Komunikat przesłany z przeglądarki do serwera WWW jest nazywany żądaniem HTTP. Gdy serwer odbierze takie żądanie, sprawdza, czy posiada odpowiednią stronę. Jeżeli znajdzie ją, dzieli na fragmenty zawarty w niej kod HTML (używając TCP), kieruje te fragmenty poprzez sieć na adres przeglądarki (za pomocą HTTP). Jeżeli serwer WWW nie znajdzie odpowiedniej strony, pobiera stronę zawierającą komunikat błędu (w tym przypadku złowrogi *Error 404: Strona nie została odnaleziona*), dzieli ją i odsyła do przeglądarki. Komunikat przesłany z serwera WWW do przeglądarki nazywamy **odpowiedzią HTTP**. Poniżej mamy schemat opisanego przed chwilą procesu.

Rysunek 2.2.



Protokół HTTP

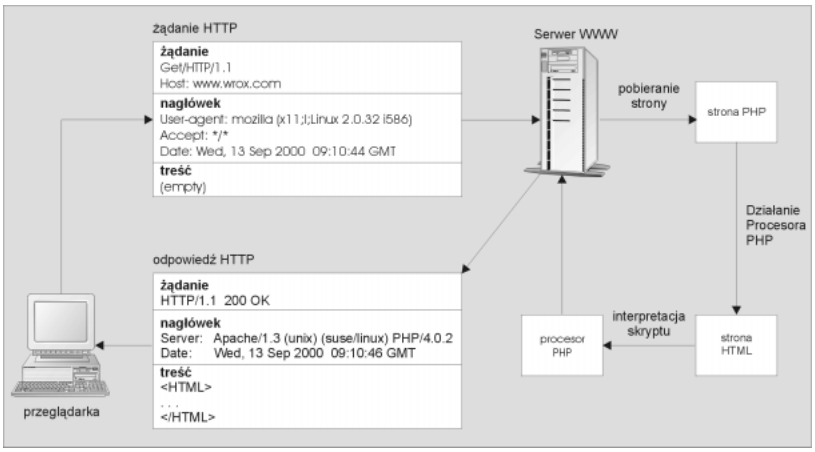
Opuściliśmy sporo szczegółów technicznych, więc teraz dokładniej opiszemy, jak działa protokół HTTP. Gdy do serwera zostanie wysłane żądanie przesłania strony WWW, zawiera ono coś więcej niż tylko interesujący nas URL. Wraz z żądaniem przesyłamy sporo dodatkowych danych. Dzieje się tak również w przypadku odpowiedzi — serwer wysyła dodatkowe dane do przeglądarki. W następnej części rozdziału zajmiemy się różnymi rodzajami dodatkowych danych przekazywanych wraz z właściwymi.

Wiele danych przesyłanych w komunikacie HTTP generuje się automatycznie i użytkownik nie ma na to bezpośredniego wpływu, więc nie musisz się przejmować ich wysyłaniem. Mimo że nie musisz ich tworzyć, są one dla nas ważne, ponieważ skrypt PHP może korzystać z tych danych i dzięki temu mamy bezpośredni wpływ na ich postać.

Każdy komunikat HTTP ma ten sam format (niezależnie, czy jest on żądaniem klienta czy odpowiedzią serwera). Możemy wyróżnić w nim trzy części: **linia żądania/odpowiedzi**, **nagłówek HTTP** i **treść HTTP**. Zawartość tych części zależy od tego, czy komunikat jest żądaniem czy odpowiedzą HTTP — oba te przypadki omówimy osobno.

Narysujmy teraz schemat tego procesu.

Rysunek 2.3.



Możemy zobaczyć, że żądanie i odpowiedź HTTP mają bardzo podobną strukturę i wiele danych jest identycznych dla obu części nagłówka HTTP. Fragmenty informacji, które przeznaczone są tylko dla serwera lub tylko dla przeglądarki są wysyłane jedynie jako część żądania bądź odpowiedzi, więc sensowne będzie dokładne przeanalizowanie ich części składowych.

Żądanie HTTP

Przeglądarka wysyła do serwera WWW żądanie HTTP, które zawiera następujące elementy.

Linia żądania

Pierwsza linia każdego żądania HTTP jest linią żądania, która zawiera następujące dane:

- komendę HTTP zwaną też **metodą**,
- ścieżkę na serwerze do zasobów żądanych przez klienta,
- numer wersji HTTP.

Przykładowa linia żądania może wyglądać następująco:

```
GET /testpage.htm HTTP/1.1
```

Używana metoda wskazuje serwerowi, w jaki sposób ma obsłużyć żądanie. Istnieją trzy podstawowe metody używane w tym polu.

Tabela 2.2.

Metoda	Opis
GET	Jest to żądanie przesłania danych z określonego URL. Większość żądań HTTP w Internecie to właśnie żądania GET. Informacja zamawiana przez nie może być

	czymkolwiek, od strony HTTP lub PHP do wyniku działania programu JavaScript lub PerlScript czy dowolnego innego programu. Możesz wysłać do serwera ograniczoną ilość danych w postaci rozszerzenia adresu URL.
HEAD	Jest to taka sama metoda jak GET, poza tym, że wskazuje na żądanie tylko nagłówek HTTP bez dołączonych danych.
POST	Żądanie to wskazuje, że dane zostaną wysłane do serwera jako część treści HTTP. Dane te są następnie przesyłane do programu obsługi na serwerze WWW.

Istnieje wiele innych metod obsługiwanych przez HTTP — PUT, DELETE, TRACE, CONNECT i OPTIONS. Są one jednak dużo rzadziej stosowane i dlatego wykraczają poza ramy tej książki. Jeżeli chcesz dowiedzieć się więcej na temat tych metod, przeczytaj RFC 2068 dostępne na serwerze *http://www.rfc.net*.

Nagłówek HTTP

Następną częścią danych jest **nagłówek** HTTP. Zawiera on dane na temat typów dokumentów, jakie klient akceptuje w odpowiedzi na żądanie, typ przeglądarki wysyłającej żądanie, datę i inne dane konfiguracyjne. Nagłówek żądania HTTP zawiera dane, które mogą być zakwalifikowane do jednej z trzech kategorii.

- **Ogólne**: zawierają informacje na temat zarówno klienta, jak i serwera, ale nie są specyficzne dla żadnego z nich.
- **Jednostka**: informacje na temat danych przesyłanych pomiędzy klientem a serwerem.
- **Żądanie**: dane na temat konfiguracji klienta i różnych typów akceptowanych dokumentów.

Przykładowy nagłówek HTTP wygląda następująco:

```
Accept: */*
Accept-Language: en-us
Connection: Keep-Alive
Host: www.wrox.com
Referer: http://webdev.wrox.co.uk/books/SampleList.php?bookcode=3730
User-Agent: Mozilla (X11; I; Linux 2.0.32 i586)
```

Jak można zauważyć, nagłówek HTTP składa się z kilku wierszy. Każdy z wierszy zawiera opis fragmentu danych nagłówka HTTP oraz jego wartość.

Istnieje wiele różnych rodzajów linii, z których może składać się nagłówek HTTP, większość z nich jest opcjonalna, więc HTTP musi wskazywać, kiedy kończy przesyłanie danych nagłówka. Do tego celu używana jest pusta linia. W HTTP 1.1 żądanie musi zawierać linię żądania i nagłówek HOST.

Treść HTTP

Jeżeli w linii żądania HTTP została użyta metoda POST, treść żądania HTTP zawiera wszystkie dane wysłane do serwera, na przykład dane, które użytkownik wpisał do

formularza HTML (w dalszej części tej książki znajdują się takie przykłady). W przeciwnym przypadku treść żądania HTTP jest pusta.

Odpowiedź HTTP

Odpowiedź HTTP jest odsyłana z serwera do klienta i zawiera następujące składniki.

Linia odpowiedzi

Linia odpowiedzi zawiera tylko dwie dane:

- numer wersji HTTP,
- kod żądania HTTP, który wskazuje na udane lub nieudane wykonanie żądania.

Przykładowa linia odpowiedzi to:

```
HTTP/1.1 200 OK
```

W tym przykładzie zwracany jest kod status HTTP 200, który jest odpowiednikiem komunikatu `OK`. Oznacza on, że żądanie zostało prawidłowo przetworzone i odpowiedź zawiera odpowiednią stronę lub dane z serwera. Przypomnij sobie, że kilka stron wstecz wspominaliśmy o kodzie statusu 404 — linia odpowiedzi zawiera kod 404, jeżeli serwer nie znajdzie żądanego zasobu. Istnieje pięć klas odpowiedzi, wymienionych w tabeli 2.3.

Nagłówek HTTP

Nagłówek odpowiedzi HTTP jest podobny do opisanego przed chwilą nagłówka żądania. W przypadku odpowiedzi HTTP dane zawarte w nagłówku są trzech rodzajów.

Tabela 2.3.

Klasa kodów	Opis
100 — 199	Są to kody informacyjne — wskazują, że bieżące żądanie jest przetwarzane.
200 — 299	Kody oznaczające sukces — serwer WWW prawidłowo otrzymał żądanie i odesłał wynik.
300 — 399	Kody oznaczające, że żądanie nie może być spełnione — jest niekompletne, nieprawidłowe lub niemożliwe do wykonania.
400 — 499	Kody oznaczające błąd klienta — żądanie było niekompletne, nieprawidłowe lub niemożliwe do wykonania.
500 — 599	Kody oznaczające błąd serwera — żądanie wygląda na prawidłowe, ale serwer nie potrafi go zrealizować.

- **Ogólne**: zawierają informacje na temat zarówno klienta, jak i serwera, ale nie są specyficzne dla żadnego z nich.
- **Jednostka**: informacje na temat danych przesyłanych pomiędzy klientem a serwerem.

- **Odpowiedź**: dane na temat serwera wysyłającego odpowiedź i sposobu obsługi odpowiedzi.

Tak jak poprzednio nagłówek składa się z kilku linii i korzysta z pustej linii oznaczającej koniec danych nagłówka. Poniżej zamieszczamy przykład nagłówka, a po prawej stronie każdej linii znajduje się jej nazwa:

```
HTTP/1.1 200 OK                                          — linia statusu
Date: Mon, 1st Nov 1999, 16:12:23 GMT                    — nagłówek ogólny
Server: Apache/1.3.12 (Unix) (SUSE/Linux) PHP/4.0.2      — nagłówek odpowiedzi
Last-modified: Fri, 29th Oct 1999, 12:08:03 GMT          — nagłówek jednostki
```

Na temat pierwszej linii już pisaliśmy, na temat drugiej nie mamy nic do powiedzenia. W linii trzeciej, `Server`, określamy typ uruchomionego serwera WWW. Czwarta zawiera datę ostatniej modyfikacji strony, którą właśnie ściągamy.

> Nagłówek może zawierać wiele więcej danych niż przytoczone tutaj, w zależności od rodzaju żądanych danych. Jeżeli chcesz dowiedzieć się więcej na temat wszystkich typów informacji, jakie mogą znajdować się w trzech częściach nagłówka, znajdziesz je w dokumencie RFC 2068 (Części 4.5, 7.1 i 7.2).

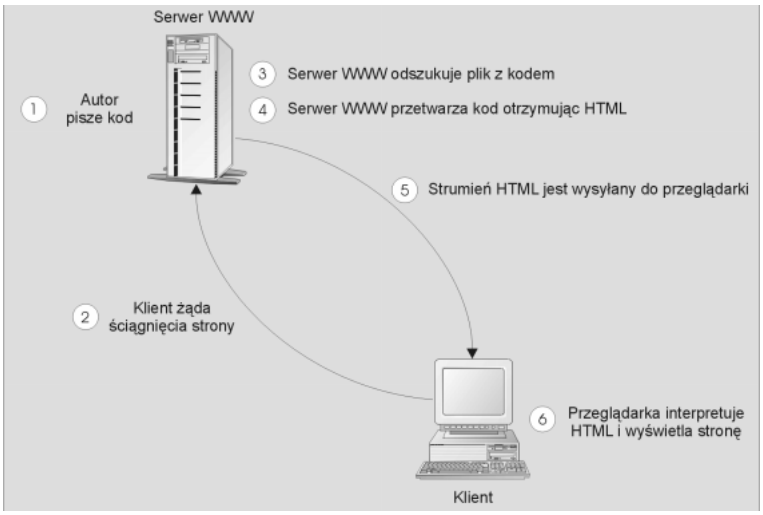
Treść HTTP

Jeżeli udała się realizacja żądania, **treść** odpowiedzi HTTP zawiera kod HTML (wraz z wszystkimi skryptami, jakie będzie mogła wykonać przeglądarka), który będzie interpretowany przez przeglądarkę.

Na scenę wchodzi PHP

Teraz wiemy już, w jaki sposób przeglądarka wysyła żądania i jak serwer WWW odsyła strony do przeglądarki. Podsumujmy 6-stopniowy proces dostarczenia strony WWW.

Rysunek 2.4.



Serwer WWW odszukuje żądaną stronę WWW (krok 3.) i, jeżeli jest to strona PHP, musi przetworzyć kod PHP w celu wygenerowania HTML, który następnie jest przesyłany do przeglądarki (krok 4.). Jeżeli nazwa strony jest zakończona *.php*, serwer przesyła ją w celu przetworzenia do maszyny skryptowej PHP (podłączonej do serwera).

Do tej pory przestudiowaliśmy wiele niezbędnych aspektów otaczających PHP, ale nasza wiedza na temat samego działania PHP jest raczej ogólnikowa. Dla uproszczenia traktowaliśmy PHP jak maszynę do produkcji parówek — serwer WWW napychał ją z jednej strony surowym kodem PHP, a z drugiej strony wychodziła ładnie zapakowana strona HTML — czas odtajnić ten proces.

Jak już wspominaliśmy, strony PHP składają się z tekstu, znaczników HTML i skryptów PHP. Programy, które wpisujesz na strony HTML, nazywane są **skryptami**. Sam HTML nie może być nazwany językiem programowania ponieważ, oprócz wyświetlania statycznego tekstu i obrazków, posiada bardzo ograniczone możliwości, dlatego niezbędne stało się pisanie kodu w innych językach, pozwalających zrealizować dodatkowe funkcje, nazywanych **językami skryptowymi**. Istnieje wiele języków skryptowych i być może znasz już kilka, na przykład JavaScript. Jednak tym, co odróżnia PHP od JavaScript czy prostego kodu HTML jest to, że jest wykonywany na serwerze, a nie w przeglądarce.

HTML pozwala na dołączanie skryptów w (prawie) dowolnym miejscu kodu HTML — robi to, dostarczając standardowych sposobów dołączania skryptów, o czym za chwilę. Gdy serwer otrzyma żądanie wysłania strony i generowany będzie kod HTML, każdy skrypt na stronie jest przesyłany do **maszyny skryptowej** w celu interpretacji i wygenerowania odpowiedniego kodu HTML.

Skrypty są podstawą PHP! Używamy ich do napisania instrukcji pozwalających na dynamiczne tworzenie stron. Istnieje wiele zalet dynamicznego tworzenia stron, można prezentować informacje wygenerowane w oparciu o dane przekazane w formularzu, można zmieniać postać stron dla różnych przeglądarek, można personalizować strony i

korzystać z ustawień użytkownika oraz wiele, wiele innych. Wszystko jest oparte na tym, że skrypt musi zostać zinterpretowany w czasie, gdy zostanie wywołany.

Koncepcja analizy i wykonania (krótka dygresja)

Interpretacja skryptów PHP jest podzielona na dwa podprocesy. Najpierw sprawdzana jest poprawność skryptu PHP w procesie zwanym **analizą**. Jest to odpowiednik sprawdzania gramatyki zdań i odszukiwania pomyłek przy pisaniu. Nie zapewnia to, że skrypt PHP jest poprawny, sprawdzana jest tylko jego zgodność ze zbiorem predefiniowanych zasad. Drugim w kolejności jest proces **wykonywania** skryptu. Tutaj okazuje się, czy napisany kod ma sens. W czasie wykonania pobierana jest pojedyncza linia PHP i zamieniana na odpowiadający jej kod HTML. Jest przeprowadzane liniowo, chyba że zostanie to zmienione w skrypcie PHP.

W PHP istnieją dwa miejsca, w których mogą być zwracane błędy: w czasie analizy i w czasie wykonania. Jeżeli zdarzy się taka sytuacja, komunikat błędu zostanie wyświetlony w przeglądarce. Jeżeli wszystko pójdzie dobrze, do przeglądarki trafi dynamicznie utworzona strona HTML.

Skrypty serwera

W jaki sposób zmienia się sposób przetwarzania strony na serwerze po wprowadzeniu skryptów? W większej części naszego modelu według którego przeglądarka łączy się do serwera WWW, wysyłania żądania, uzyskuje odpowiedź i interpretuje otrzymany HTML tak, aby uzyskać wynikową stronę, nie zachodzą żadne zmiany.

Jedyna zmiana zachodzi w czasie przygotowania strony wysyłanej do przeglądarki, gdy serwer napotka na skrypt. Pierwszym zadaniem serwera jest identyfikacja maszyny odpowiedzialnej za interpretację tego skryptu. Jest to ważny punkt, ponieważ w czasie pisania skryptu wybieramy, czy zostanie on wykonany na serwerze lub w przeglądarce. Sprecyzujmy te różnice.

- Skrypt interpretowany przez serwer WWW jest nazywany **skryptem serwera**. Jest to zbiór instrukcji przetwarzanych przez serwer i generujących HTML. Wynikowy HTML jest wysyłany jako część odpowiedzi HTTP do przeglądarki. Przeglądarka analizuje HTML i odpowiednio go wyświetla.
- Strona HTML jest interpretowana przez przeglądarkę. HTML nie jest przetwarzany przez serwer WWW. Jest wysyłany do przeglądarki (jako część odpowiedzi HTTP) i tam interpretowany. Wyniki interpretacji widzimy w oknie przeglądarki.

Identyfikacja skryptów PHP

Do tej pory powiedzieliśmy, że serwer WWW musi zdecydować, na którym komputerze będzie przetwarzany skrypt. Jednak w jaki sposób mamy zidentyfikować skrypt wbudowany w kod HTML? Tak naprawdę w tym rozdziale odpowiedzieliśmy już na to

pytanie — PHP (który ma być przetwarzany na serwerze WWW) jest zawarty pomiędzy specjalnymi znacznikami <?php ... ?>, jak w następującym przykładzie:

```
Od początku tego roku minęło
<?php echo gmDate("z"); ?> dni
```

Wszystko, co jest zawarte pomiędzy <?php a ?>, jest traktowane jako kod PHP i wysyłane do maszyny skryptowej PHP w celu przetworzenia. Po interpretacji do przeglądarki zwracany jest HTML i, jeżeli skorzystasz z opcji przeglądarki *Źródło*, powinieneś zobaczyć, że nasz program zwrócił czysty kod HTML:

```
<HTML>
<BODY>
Od początku tego roku minęło
263 dni
</BODY>
</HTML>
```

W ten sposób uniknięto tradycyjnego problemu możliwości przeglądarki. Szczególnie opcje, takie jak komponenty wbudowywane za pomocą znaczników <OBJECT>, działają jedynie w programie Internet Explorer. Poza tym, działanie nowych funkcji, takich jak arkusze stylów, zależało od możliwości poszczególnych przeglądarek, na przykład IE3 bardzo słabo obsługiwał arkusze stylów, IE4 robił to już w szerokim zakresie, a IE5 posiada wreszcie kompletną obsługę.

PHP nie stwarza tej kategorii problemów — nie polegasz na tym, że przeglądarka potrafi obsługiwać PHP, przeglądarka musi jedynie przetwarzać HTML. Jedynym problemem jest sposób interpretacji różnych funkcji HTML w przypadku używania starszych wersji programów Netscape Navigator, Internet Explorer lub Opera. W przypadku najnowszych wersji problem nie występuje.

Pamięć podręczna

Zanim przejdziemy do programowania w PHP, powinniśmy wspomnieć o pamięci **podręcznej przeglądarki**. Prawdopodobnie znasz tę opcję, przeglądane strony są zapisywane na dysku. Aby przyspieszyć przeglądanie stron, przeglądarka korzysta z tych stron w przypadku, gdy na serwerze nie ma nowszej wersji strony. Miejsce ich zapisu na komputerze zależy zarówno od typu używanej przeglądarki, jak i platformy, na której korzystasz z przeglądarki.

W Windows (98, NT i 2000) pamięć podręczna Netscape Navigatora jest umieszczona zwykle w katalogu *C:\Program Files\Netscape\Users\Cache*. W przypadku Internet Explorera jest to katalog *Temporary Internet Files* w katalogu *Windows* lub *WinNt*, w zależności od tego, czy korzystasz z Windows 98 lub NT 4.0. W Windows 2000 katalog ten znajduje się w *Documents and Settings* w osobnym podkatalogu dla każdego użytkownika korzystającego z systemu.

W systemie Linux pamięć podręczna znajduje się w katalogu *~/.netscape/cache* (znak ~ reprezentuje katalog domowy użytkownika).

W przypadku używania PHP na serwerze do przetwarzania stron, istnienie pamięci podręcznej staje się znaczące. Jeżeli wykorzystasz zapamiętaną stronę z poprzedniego wykonania, nie skorzystasz z dynamicznego kodu HTML i zwracane dane mogą być nieprawidłowe. Możesz to zauważyć, jeżeli użyjesz przycisku przeglądarki *Wstecz*, przechodząc przez kilka stron PHP. Czasami nie pomaga nawet naciskanie przycisku *Odśwież*, bufor nie jest odświeżany i strona wyświetla się nieprawidłowo, ponieważ przeglądarka nie ma informacji o zmianach. W takim przypadku należy wymusić odświeżenie strony. W Internet Explorer 5 należy nacisnąć klawisze *Ctrl+F5*, natomiast w Netscape Navigator nacisnąć *Shift* i kliknąć przycisk *Reload*.

Jest to ważne, ponieważ w przypadku uruchamiania przykładów możesz otrzymywać te same wyniki, pochodzące z poprzedniego uruchomienia skryptu, które przeglądarka może pobrać z bufora. Jeżeli dostajesz niezrozumiałe wyniki upewnij się, że przeglądarka pobiera nową stronę. Można to zrobić, wymuszając odświeżenie lub zmieniając ustawienia przeglądarki.

Istnieją nagłówki HTTP używane do zabezpieczania strony przed przechowywaniem w pamięci podręcznej. Poniższe 3 linie kodu PHP (jeżeli użyjesz ich **na początku** skryptu) zabezpieczają przed zapamiętaniem strony przez Internet Explorer lub Netscape Navigator. Można posłużyć się tym sposobem, aby zabezpieczyć się przed nieprzewidzianymi wynikami lub w przypadku, gdy jesteś zmęczony wielokrotnym odświeżaniem dokumentu.

```
<?
header("Cache-Control: no-cache, must-revalidate");
header("Pragma: no-cache");
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
?>
```

Zmienne

Po zapoznaniu się ze sposobem umieszczania skryptów PHP w stronach WWW oraz tym, w jaki sposób serwer WWW je obsługuje, należy się zająć poznawaniem różnych elementów języka PHP. Chyba najbardziej podstawową jednostką w programowaniu jest **zmienna**, zwykle określana jako obszar pamięci służący do przechowywania danych, który posiada identyfikator nadany mu przez programistę.

W PHP zmienne rozpoczynają się znakiem `$`. W celu przypisania wartości zmiennej używamy operatora przypisania, czyli znaku `=`. Aby utworzyć zmienną w PHP i przypisać jej wartość, użyjemy następującej linii:

```
$autor = "William Shakespeare";
```

Identyfikatorem zmiennej jest `$autor`, natomiast przypisaną wartością jest tekst `"William Shakespeare"`. Można również do zmiennych przypisywać liczby.

```
$ilosc_cyfr_w_jednej_dloni = 5;
```

Mamy tutaj złożoną nazwę zmiennej `$ilosc_cyfr_w_jednej_dloni`, ale oprócz tego nie różni się ona zbytnio od poprzedniego przypisania. Jedyną różnicą jest to, że zmienne numeryczne nie są objęte cudzysłowami. Wskazuje to PHP, że mają być traktowane jako wartość numeryczna. Nie ma żadnych innych wymagań, aby PHP traktowało wartość jako liczbę.

Po utworzeniu zmiennej możesz jej dowolnie używać w programie. W celu wyświetlenia wartości na ekranie możesz użyć naszej znajomej funkcji `echo()` w następujący sposób:

```
echo $autor;
```

Ograniczenia w nazwach zmiennych

Istnieje niewiele ograniczeń w nazywaniu zmiennych. We wielu językach programowania istnieje limit długości nazwy zmiennej, zwykle do 255 lub 1000 znaków, jednak w PHP nie ma takiego ograniczenia. Prawdopodobnie nigdy nie będziesz używał powyżej 50 znaków, a 20 do 30 jest wystarczające dla większości ludzi.

Pierwszym ograniczeniem jest to, że zmienna musi zaczynać się od litery lub podkreślenia (ignorując znak dolara, który nie jest właściwie częścią nazwy zmiennej). Drugim jest to, że nazwa może składać się z liter, cyfr i znaków podkreślenia. Inne znaki, takie jak +, -, *, i & nie są dopuszczalne i powodują wypisanie błędu na stronie. Oprócz tego możesz tworzyć dowolne nazwy według własnego uznania i fantazji. Jednak w dalszej części tego rozdziału zajmiemy się rozsądnym nazywaniem zmiennych.

Wielkość liter w nazwach zmiennych

Zmienne nie są tak proste, jak się to wydaje na pierwszy rzut oka. Jednym z problemów, z którym początkujący spotykają się najczęściej, jest rozpoznawanie wielkich i małych liter w nazwach zmiennych. Najłatwiejszym sposobem rozwiązania jest użycie przykładowego kodu:

```
$autor = "William Shakespeare";
$Autor = "James Joyce";
```

Powyższe dwie linie tworzą dwie różne zmienne, jedną o nazwie `$autor` i drugą o nazwie `$Autor`. Te dwie zmienne mają dwie różne wartości i są tak samo różne jak te, których użyjemy w poniższym kodzie:

```
$slawny_autor_angielski = "William Shakespeare";
$slawny_autor_irlandzki = "James Joyce";
```

Dosyć często zdarza się, że przy wypisywaniu wartości zmiennej na stronie przypadkowo użyjemy pierwszej wielkiej litery tam, gdzie jej nie powinno być. Jeżeli masz zamiar skorzystać z jednej tylko zmiennej, powinieneś użyć następującego kodu:

```
$autor = "William Shakespeare";
$autor = "James Joyce";
```

Pierwsza linia ustawia wartość zmiennej `$autor` na `"William Shakespeare"`, natomiast druga zmienia całkowicie zawartość `$autor` na `"James Joyce"`. W tym przykładzie kodu utworzona i używana jest tylko jedna zmienna.

Teraz zajmiemy się małym przykładem, w którym tworzymy zmienną i wyświetlamy jej wartość na stronie WWW.

Przypisywanie wartości do zmiennej i jej wyświetlanie

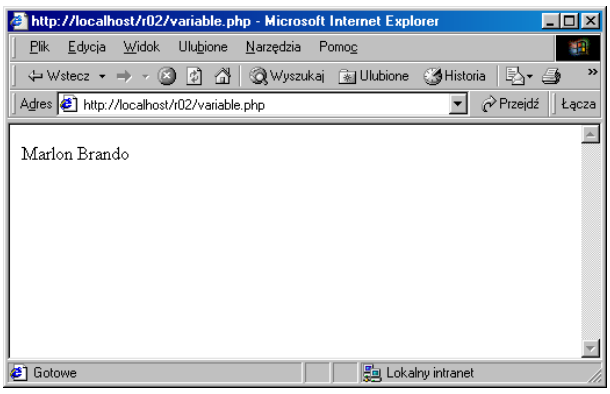
1. Otwórz ulubiony edytor tekstu i wpisz następujące linie:

```
<HTML>
<BODY>
<?php
$actor = "Marlon Brando";
echo $actor;
?>
</BODY>
</HTML>
```

2. Zapisz ten kod jako *variable.php*.

3. Otwórz tę stronę w przeglądarce.

Rysunek 2.5.



Jak to działa?

Właściwy kod składa się jedynie z dwóch linii. W pierwszej

```
$actor = "Marlon Brando";
```

przypisujemy ciąg `"Marlon Brando"` do zmiennej `$actor`. W drugiej przekazujemy zawartość zmiennej `$actor` do funkcji `echo()`.

```
echo $actor;
```

Jak widać, wysyłając zawartość zmiennej `$actor` na stronę WWW, nie używaliśmy cudzysłowów. Jeżeli jednak użyjemy takiej linii:

```
echo "$actor";
```

na ekranie nadal będziemy widzieli napis `Marlon Brando`. Aby PHP nie podstawiało wartości zmiennych, musisz zamiast cudzysłowów skorzystać ze znaków apostrofu.

> Mając doświadczenie tylko z Windows, poczujesz się nieco zagubiony. Możesz spodziewać się, że powyższa linia wyświetli tekst `$actor`. Autorzy, pisząc PHP, zapożyczyli niektóre elementy Perla i skryptów powłoki Unix, a w tych skryptach użycie znaku dolara „nadpisuje" znaki cudzysłowów. Aby wyświetlić znak dolara z następującą po nim literą (lub literami), musisz stosować inne metody, którymi zajmiemy się w kolejnych rozdziałach.

Typy danych

Jak wspominaliśmy wcześniej, skupimy się na dwóch typach zmiennych, zawierających wartości numeryczne i ciągi znaków. Istnieje jeszcze kilka innych typów zmiennych używanych przez PHP, nazywanych **typami danych**, takich jak tablice, obiekty, jednak ich szczegółowy opis znajdzie się w późniejszych rozdziałach. Pełna lista typów danych w PHP jest następująca:

- `string` (tekst),
- `integer` (liczby),
- `double` (liczby),
- `array` (tablice),
- `object`,
- typ nieokreślony.

Typy danych nie są ustawiane przez programistę, to PHP decyduje za ciebie, w trakcie interpretacji strony, jaki powinien być typ zmiennej i odpowiednio go ustawia. Te typy danych, używane przez PHP, oznaczają różne rodzaje przechowywanych danych i definiują różne zestawy operacji, jakie można na tych danych wykonać.

Znakowy typ danych

Typ danych `string` jest jedynym, który przechowuje dane tekstowe, słowa lub całe zdania. Wszystko, treść zapisana w cudzysłowach, jest traktowana jako tekst, nawet dane numeryczne, na przykład w poniższym kodzie obie zmienne przechowują ciągi:

```
$CarType = "Cadillac";
$EngineSize = "2.6";
```

Nie ma znaczenia, że druga z wartości jest czysto numeryczna, jeżeli została podana w cudzysłowach, automatycznie staje się ciągiem. Jeżeli jakaś wartość zostanie zapisana w postaci ciągu, a następnie wykonasz na niej operacje matematyczne, PHP dokona konwersji typu zmiennej na typ numeryczny. W wyniku tego dodawanie ciągów, które

mają wartość numeryczną, może doprowadzić do nieoczekiwanych wyników. Ciągi mają własną operację dodawania, jest nazywana **łączeniem** (konkatenacją).

Łączenie ciągów

Łączenie ciągów to proces dodawania jednego ciągu do drugiego, to znaczy dołączania jednego ciągu na końcu drugiego. Do tego celu używa się operatora złączenia `.` (kropka). Wynikiem działania poniższej linii jest ciąg `Cadillac2.6`, zakładając wartości zmiennych, takie jak w poprzednim przykładzie.

```
$Car = $CarType . $EngineSize;
```

W naszym przykładzie nie ma odstępu pomiędzy łączonymi ciągami, co może nie podobać się projektantom. Utwórzmy więc zmienną, która będzie przechowywała znak spacji.

```
$Space = " ";
```

Powinieneś zauważyć, że różni się ona nieco od pustego ciągu. Pusty ciąg nie zawiera nic, natomiast ciąg ze spacją zawiera jeden znak. Spacje mogą być oczywiście łączone z innymi ciągami w taki sam sposób, jak inne teksty.

```
$Car = $CarType . $Space . $EngineSize;
```

Wykonanie tej linii da w wyniku porządnie wyglądający tekst `Cadillac 2.6`.

Nie ma powodu, dla którego nie mógłbyś łączyć zmiennych z tekstem.

```
$Car = "Buick" . $Space . "2.0";
```

W tym przykładzie dodaliśmy do tekstu zmienną i jeszcze jeden fragment tekstu, co w wyniku daje `Buick 2.0`. Można również zastosować następującą kombinację:

```
$Car = "Buick" . " " . "2.0";
```

Da to efekt identyczny z poprzednim. Należy jedynie pamiętać o zasadzie stosowania odstępów w HTML. Jeżeli w tekście będzie użyta więcej niż jedna spacja pod rząd, przeglądarka wyświetli na stronie WWW tylko jedną.

W PHP mamy jeszcze inną metodę łączenia ciągów — za pomocą funkcji `echo()`. Pamiętasz, jak próbowaliśmy wypisać nazwę zmiennej na stronie WWW i zamiast tego otrzymaliśmy wartość zmiennej, na przykład:

```
$CarType = "Cadillac";
echo "$CarType";
```

Na ekranie otrzymamy `Cadillac`. Daje to możliwość używania nazw zmiennych razem z tekstem, aby w wyniku otrzymać połączenie tych tekstów.

```
echo "$CarType Andrzeja";
```

Linia ta spowoduje wypisanie ciągu `Cadillac Andrzeja`. Istnieją jednak pułapki, o których należy pamiętać. Po pierwsze, znaków sterujących można używać pomiędzy

cudzysłowami, a nie wolno między apostrofami. Należy pomyśleć co się stanie, jeżeli będziesz miał dwie zmienne, na przykład `$Car` i `$Cars`.

Co będzie wynikiem działania poniższej linii?

```
echo "Kliknij tu aby przejść do $Carsale";
```

To może być dla ciebie oczywiste — chcesz użyć zmiennej `$Car`, a nie `$Cars`, jednak w jaki sposób zasygnalizować to PHP? Tak naprawdę PHP będzie szukało zmiennej `$Carsale`. Aby uzyskać interesujące nas działanie, należy ująć nazwę zmiennej w nawiasy klamrowe.

```
echo "Kliknij tu aby przejść do ${Car}sale";
```

Po drugie, trzeba również pamiętać o oddzielaniu spacją kropki od zmiennej. Musimy to robić nie tylko dla lepszej czytelności, może to mieć również wpływ na wynik. Dwie linie poniżej mogą wyglądać na identyczne, ale dają zupełnie inne wyniki.

```
echo 2 . 2;
echo 2.2;
```

Wynik to odpowiednio 22 i 2.2! Trochę oszukiwaliśmy, ponieważ żadna z cyfr nie była umieszczona w apostrofach, więc nie są to tak naprawdę ciągi. Jednak problem nadal występuje, ponieważ liczba 2 została potraktowana przez operator złączenia jak ciąg znaków. Aby PHP potraktowało liczbę jak ciąg, wystarczy dodać odstęp pomiędzy tą liczbą a operatorem złączenia. Należy zapamiętać, że w niektórych przypadkach sposób umieszczania odstępów w kodzie może wpłynąć na wygląd tworzonej strony.

Czas zobaczyć w działaniu niektóre z omówionych możliwości PHP. Utwórzmy kilka ciągów wraz z ich wartościami i przy użyciu funkcji `echo()` wyświetlmy je na stronie WWW.

Użycie ciągów znaków

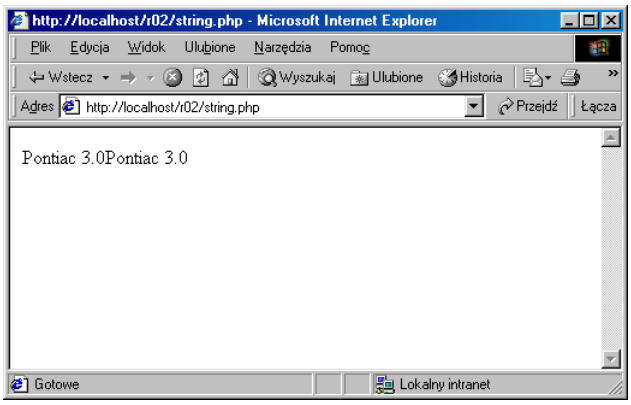
1. Otwórz edytor stron WWW i wpisz następujący kod:

```
<HTML>
<BODY>
<?php
$CarType = "Pontiac";
$EngineSize = "3.0";
$Space = " ";
$Car = $CarType . $Space . $EngineSize;
echo $Car . $Car;
?>
</BODY>
</HTML>
```

2. Zapisz go jako *string.php*.

3. Po otwarciu w przeglądarce powinieneś otrzymać następujący wynik.

Rysunek 2.6.



Jak to działa?

Zajmowaliśmy się już podobnym fragmentem kodu. Pierwsze trzy linie tworzą trzy zmienne `$CarType`, `$EngineSize` oraz `$Space` i przypisywane są im różne wartości.

```
$CarType = "Pontiac";
$EngineSize = "3.0";
$Space = " ";
```

W następnej linii łączymy te trzy wartości, a wynik umieszczamy w nowo utworzonej zmiennej o nazwie `$Car`.

```
$Car = $CarType . $Space . $EngineSize;
```

Na koniec możemy dołączyć ten wynik do samego siebie w funkcji `echo()`.

```
echo $Car . $Car;
```

Skutkiem tego wynik jest dwukrotnie powtórzony, ale nie ma odstępu pomiędzy dwoma zmiennymi `$Car`. Wygląda to niezbyt ładnie, ale chcieliśmy w ten sposób pokazać, że jest możliwe łączenie zmiennej z samą sobą.

To nie działa!

Jeżeli zobaczysz tekst podobny do *Parse error in C:\Program Files\Apache Group\Apache\htdocs\string.php on line n*, prawdopodobnie zapomniałeś umieścić średnik na końcu linii. Jeżeli widzisz jedynie część wyniku pokazanego na rysunku, prawdopodobnie pomyliłeś się w nazwie zmiennej lub zmieniłeś wielkość jednej z liter nazwy zmiennej. Upewnij się, że nazwy i wielkości liter są poprawne dla wszystkich zmiennych użytych w tym przykładzie. Jeżeli tak się stało, przypadkowo utworzyłeś nową pustą zmienną.

Numeryczne typy danych

Istnieją dwa numeryczne typy danych, `integer` i `double`. Typ `integer` to liczby całkowite, natomiast `double` to liczby zmiennoprzecinkowe. Poniżej mamy kilka przykładów ich użycia.

```
$liczba_integer = 33;
$inny_integer = -5797;

$liczba_double = 4.567;
$inna_double = -23.2;
```

Powinieneś wydedukować z powyższych linii, że każda liczba całkowita staje się automatycznie liczbą `integer`, natomiast liczby z ułamkami są zapamiętywane jako typ `double`.

Te dwa typy danych mają również różne zakresy wartości, które zależą jednak od systemu operacyjnego i platformy, na której działa PHP, na przykład PHP na Windows 98 może przechowywać liczby `integer` od –32768 do 32767. Typ `double` na tej samej maszynie ma zakres od –1.79769313486232E308 do –4.940656458412E-324 (dla liczb ujemnych) i 4.940656458412E-324 do 1.79769313486232E308 (dla liczb dodatnich) oraz wartość zero.

> Notacja E (wykładnicza) przytoczona tutaj jest metodą, której PHP i praktycznie każdy inny język programowania używa do zapisu bardzo małych i bardzo dużych liczb. Liczba jest zapisywana jako liczba bazowa pomnożona przez dziesięć podniesione do potęgi określonej drugą liczbą. Na przykład 2.5E3 to zapis $2.5*10^3$ lub 2500.

To, co różni PHP od innych języków programowania, to o wiele mniejsza ilość typów danych. Istnieją tylko dwa typy numeryczne. Upraszcza to tworzenie programów i zwykle łatwo jest określić, kiedy potrzebujemy liczb całkowitych, a kiedy liczb z przecinkiem.

Proste operacje matematyczne

PHP posiada zestaw operatorów służących do wykonywania operacji matematycznych. Są to bardzo popularne operatory, z którymi powinieneś się zapoznać już na matematyce w szkole.

Operator	Działanie
+	Operator dodawania.
*	Operator mnożenia.
-	Operator odejmowania, a także operator jednoargumentowy wskazujący liczby ujemne, na przykład –6.
/	Operator dzielenia.
%	Operator modulo (reszta z dzielenia całkowitego), na przykład, 8 % 5 = 3.

Ich użycie jest całkowicie intuicyjne. Dla przykładu użyjemy operatora dodawania do podliczenia wartości paragonu w sklepie.

```
$Chleb = 1.5;
$Mleko = 0.8;
$Suma = $Chleb + $Mleko;
```

Wartość zakupów jest sumą wartości zmiennej `$Chleb` i `$Mleko`. Po wykonaniu operacji zmienna `$Suma` ma wartość `2.3`. Inne operatory działają w ten sam sposób. Jeżeli chciałbyś kupić dwa bochenki chleba, kod obliczający wartość zakupów jest następujący:

```
$Chleb = 1.5;
$Suma = $Chleb * 2;
```

Można łączyć operatory matematyczne w wyrażenia w sposób pokazany w kolejnym przykładzie.

```
$Chleb = 1.5;
$Mleko = 0.8;
$KuponRabatowy = 0.5;
$Suma = $Chleb + $Mleko - $KuponRabatowy;
```

Dodawanie zmiennej do siebie samej

Jedną z operacji dozwoloną w PHP, która może wprawić w zakłopotanie matematyków, jest:

```
$Suma = $Suma + $Chleb;
```

Nie oznacza to, że `$Suma` jest równa `$Suma + $Chleb`, z czego wynika, że `$Chleb` jest równy `0`. Znak równości w tym przypadku jest operatorem przypisania, a wyrażenie to oznacza, że do zmiennej `$Suma` jest przypisywana STARA wartość `$Suma` powiększona o wartość zmiennej `$Chleb`. Poniższa linia

```
$Suma = $Suma + 1;
```

powoduje powiększenie o jeden wartości zmiennej `$Suma`. Istnieje również skrócony zapis operacji zwiększenia o jeden.

```
$Suma++;
```

Efekt działania tej linii jest identyczny jak poprzedniej. Innym skróconym operatorem jest:

```
$Suma += 2;
```

Można nawet wykonać następującą operację:

```
$Suma += $Suma;
```

Wypróbuj ją i sprawdź, jaki jest wynik tego wyrażenia.

W PHP istnieje więcej operatorów matematycznych oprócz tych, które opisaliśmy, na przykład istnieje zestaw funkcji do wykonywania obliczeń trygonometrycznych lub logarytmicznych. Nie będziemy zajmować się nimi szczegółowo. Pełną listę funkcji matematycznych znajdziesz w dodatku B. W kolejnych rozdziałach będziemy zajmować się operatorami porównania, logicznymi i bitowymi — ich stosowanie wymaga znajomości funkcji, które musimy jeszcze poznać. Na razie znamy wystarczającą ilość operatorów.

Priorytety

Proste operacje matematyczne rządzą się tymi samymi zasadami kolejności wykonywania działań, jakich używamy w matematyce. Poniższe wyrażenie może dać różne wyniki w zależności od kolejności wykonywania operacji.

```
$Suma = 5+3*6;
```

Jeżeli obliczymy je w takiej kolejności, w jakiej jest zapisane, otrzymamy wynik 48. Jednak, stosując matematyczną kolejność wykonywania operacji, w której mnożenie wykonywane jest przed dodawaniem, otrzymasz 23. Jasne jest, że musisz mieć dostęp do metody pozwalającej określić kolejność wykonywania operacji.

Tak jak w matematyce, PHP używa nawiasów do wymuszenia kolejności wykonywania operacji. Jeżeli chcesz się upewnić, że w ostatnim przykładzie dodawanie będzie wykonane przed mnożeniem, musisz dodać nawiasy:

```
$Suma = (5+3)*6;
```

Zajmiemy się teraz przykładem, w którym użyjemy kilku operatorów i po wykonaniu obliczeń wyświetlimy wynik na stronie WWW. W tym przykładzie na wejściu mamy wynagrodzenie brutto i liczymy kwotę po odjęciu 20% podatku. Po odjęciu podatku, odejmujemy 3% na fundusz emerytalny i wyświetlamy na tej samej stronie końcową płacę netto oraz kwotę przed potrąceniem funduszu emerytalnego.

Typy numeryczne

1. Otwórz edytor i wpisz następujący kod strony:

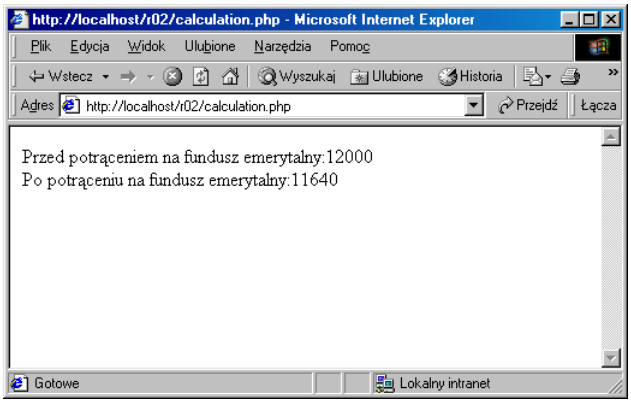
```
<HTML>
<BODY>
<?php
$Salary = 15000;
$TaxRate = 20;
$Pension = 3;
$BeforePensionIncome = $Salary - (($Salary / 100) * $TaxRate);
$AfterPensionIncome = $BeforePensionIncome -
(($BeforePensionIncome/100)*$Pension);
echo "Przed potrąceniem na fundusz emerytalny:$BeforePensionIncome<BR>";
echo "Po potrąceniu na fundusz emerytalny:$AfterPensionIncome";
?>
</BODY>
```

```
</HTML>
```

2. Zapisz jako *calculation.php*.

3. Otwórz stronę w przeglądarce. Powinieneś otrzymać następujący wynik.

Rysunek 2.7.



Jak to działa?

Kod jest dosyć prosty. W pierwszej linii tworzymy zmienną z pensją i ustawiamy ją na `15000`:

```
$Salary = 15000;
```

W drugiej linii tworzymy zmienną `$TaxRate` i nadajemy jej wartość `20`:

```
$TaxRate = 20;
```

W trzeciej linii tworzymy zmienną `$Pension` i ustawiamy ją na `3`:

```
$Pension = 3;
```

Teraz jesteśmy gotowi do przeprowadzenia obliczeń. Aby obliczyć ilość otrzymywanych pieniędzy, musimy wyliczyć, ile wynosi 20% podatku. Realizujemy to przez podzielenie zmiennej `$Salary` przez 100 i pomnożenie wyniku przez 20. Daje to w wyniku nasze 20%, które musimy odjąć od wartości początkowej.

Aby upewnić się, że 20% będzie obliczone wcześniej, otoczymy te obliczenia nawiasami. Teraz możemy odjąć je od bazowej kwoty i przypisać wynik do zmiennej `$BeforePension Income`.

```
$BeforePensionIncome = $Salary - (($Salary / 100) * $TaxRate);
```

Kolejnym krokiem jest obliczenie 3% z wartości po odjęciu podatku, przechowywanej w zmiennej `$BeforePensionIncome`. Wykorzystamy do tego dokładnie te same operacje, co przed chwilą, ale skorzystamy z innych zmiennych. Tym razem, aby uzyskać 3% podzielimy kwotę bez podatku przez 100 i pomnożymy przez 3. Teraz odejmiemy wynik od

wartości po odjęciu podatku (`$BeforePensionIncome`) i zapiszemy w zmiennej `$AfterPension Income`.

```
$AfterPensionIncome = $BeforePensionIncome -
                    (($BeforePensionIncome/100)*$Pension);
```

Po obliczeniu obu kwot możemy wyświetlić je na ekranie. Poniższe dwie linie wyświetlają tekst `"Przed potrąceniem na fundusz emerytalny"`, po którym następuje wartość zmiennej `$BeforePensionIncome`. Zauważ, że w funkcji `echo()` umieściliśmy znacznik HTML. Bardzo łatwo jest zapomnieć, że PHP generuje HTML, więc pamiętajmy, aby w tekście zamieszczać znaczniki HTML wymuszające nową linię, zamiast jedynie znaków końca linii `CR` lub `CRLF` — jest to bardzo często stosowana sztuczka. Znacznik przejścia do nowej linii `<BR>` jest interpretowany tak samo, jak w zwykłej stronie HTML i przenosi wyświetlanie drugiego wyniku do następnej linii.

```
echo "Przed potrąceniem na fundusz emerytalny:$BeforePensionIncome<BR>";
echo "Po potrąceniu na fundusz emerytalny:$AfterPensionIncome";
```

Należy również pamiętać o umieszczaniu średników na końcu każdej linii. Bez nich program spowoduje wygenerowanie komunikatu błędu.

Stałe

Do tej pory zajmowaliśmy się obiektami, które mogliśmy zmienić po przypisaniu wartości. W poprzednich przykładach utworzyliśmy zmienną `$Autor` zainicjowaną wartością `"William Shakespeare"`, która mogła być zmieniana w dowolnym momencie, na przykład na `"Herman Melville"`.

```
$Autor = "William Shakespeare";
echo $Autor;
$Autor = "Herman Melville";
echo $Autor;
```

Jeżeli umieścisz powyższe linie na stronie WWW, to zobaczysz, że funkcje `echo()` dadzą w wyniku dwa różne napisy, jeden za drugim, w kolejności zmiany wartości zmiennej.

```
William Shakespeare
Herman Melville
```

Co zrobić, jeżeli nie chcemy, aby możliwa była zmiana wartości? Niektóre wartości nie wymagają modyfikacji.

```
$TemteraturaZamarzania = 0;
$DzienNiepodleglosci = "4 lipca";
$PierwszyPrezydent = "George Washington";
```

Możesz chcieć również zdefiniować wartości, które nie są bezwzględnie niezmienne, ale nie chcemy ich zmieniać.

```
$UlubionaDruzynaNFL = "Eagles";
```

W tym przypadku nie chcesz, aby ktoś inny zmieniał wartości w kodzie, powodując powstawanie błędów. PHP posiada specjalne udogodnienie, które pozwala na tworzenie identyfikatorów, których wartości nie mogą być zmieniane. Identyfikatory te nazywają się **stałymi**. Pozwalają tworzyć zmienne i przypisywać do nich wartości, których nie można zmienić. Sposób tworzenia stałych jest nieco inny od normalnego tworzenia zmiennych.

Słowo kluczowe define

Do definicji stałych potrzebne jest specjalne słowo kluczowe `define`. Nie są one również poprzedzane znakiem dolara. W celu utworzenia stałej dla zmiennej `Temperatura Zamarzania`, musimy użyć `define` w sposób pokazany poniżej.

```
define("TEMPERATURAZAMARZANIA", 0);
```

Przyjęte jest, aby nazwy stałych zapisywane były wielkimi literami. Pierwsza wartość jest nazwą stałej, natomiast druga, wartością stałej. Aby utworzyć stałe zawierające wartości tekstowe, należy umieścić wartość stałej w cudzysłowach.

```
define ("DZIENNIEPODLEGLOSCI", "4 lipca");
define ("PIERWSZYPREZYDENT", "George Washington");
```

Stałe mogą być używane w identyczny sposób jak zmienne. Możemy więc wyświetlić jej wartość na stronie WWW za pomocą funkcji `echo()`.

```
echo "Dzień niepodległości jest " . DZIENNIEPODLEGLOSCI;
```

Możemy dodawać stałe do tekstu, używając operatora złączenia. Jednak istnieje jedna różnica wynikająca z tego, że stałe nie są poprzedzone znakiem dolara. PHP nie może odróżnić ich od zwykłego tekstu w procesie zamiany nazw zmiennych na ich wartości.

```
echo "Dzień niepodległości jest DZIENNIEPODLEGLOSCI";
```

Powyższa linia spowoduje wyświetlenie napisu `"Dzień niepodległości jest DZIEN NIEPODLEGLOSCI"`. Aby zapewnić prawidłowe wyświetlanie wartości zmiennych, musisz upewnić się, że nazwa stałej zawsze występuje poza cudzysłowami.

PHP posiada również kilka wbudowanych stałych używanych do odczytania niektórych danych, na przykład systemu operacyjnego, na jakim działa PHP lub wersji PHP i tak:

```
echo PHP_OS;
```

zwróci nazwę systemu operacyjnego, na jakim działa serwer.

Inicjalizacja

Każdy, kto zna inne języki programowania poza PHP, powinien w tym miejscu poczuć się nieco zdezorientowany. Omówiliśmy już przecież zmienne, a w większości języków programowania, w przypadku zmiennych, na początku przeprowadza się proces zwany

deklaracją lub **inicjalizacją**. Proces jest podobny w wielu językach programowania, takich jak Java lub Visual Basic. Musisz w nich zadeklarować (zainicjować) zmienną przed jej użyciem. Deklaracja lub inicjalizacja polega na tym, że zanim odwołasz się do zmiennej, musisz zadeklarować chęć jej użycia, na przykład w języku Visual Basic robi się to przy pomocy słowa `Dim`:

```
Dim nowaZmienna
nowaZmienna = "Cześć"
```

Powyższy kod nie jest kodem PHP i będzie działał jedynie w Visual Basic.

W PHP nie musimy tego robić. Nie jest to wymagane, bo w rzeczywistości pierwsze użycie zmiennej powoduje jej automatyczne utworzenie.

```
$nowaZmienna = "Cześć";
```

Istnieje jednak zaleta inicjalizacji, której w PHP czasami brakuje. Wspominaliśmy wcześniej, że każda zmienna ma typ danych nadany automatycznie przez PHP w czasie przypisywania. W innych językach programowania, w trakcie deklaracji zmiennej, podajemy zwykle, jakiego typu dane przechowywane będą w tej zmiennej, na przykład w Visual Basic.

```
Dim nowaZmienna as String
nowaZmienna = "Cześć"
```

Powyższy kod nie jest kodem PHP i będzie działał jedynie w Visual Basic.

W PHP nie można wykonać takiej operacji. Może to prowadzić czasami do problemów — co będzie, jeżeli PHP nada zmiennej typ danych, którego nie chcesz? Co stanie się, jeżeli będziesz chciał zmienić typ danych w trakcie obliczeń? Na przykład możesz pobierać od użytkownika jakieś dane w rodzaju marki i wielkości silnika jego samochodu, załóżmy, że jest to Volkswagen Golf 2.0CL. Powinny być one traktowane jak tekst. Następnie chcemy wyciąć z tego napisu wielkość silnika i użyć jej jako liczby w obliczeniach matematycznych. Możesz wymyślić wiele takich przypadków.

W podobnych sytuacjach musisz umieć konwertować typy danych zmiennej.

Konwersje

PHP posiada spory zestaw wbudowanych funkcji pozwalających na konwersje typów danych oraz funkcji, które pozwalają zorientować się, jaki typ został nadany zmiennej. Tak naprawdę PHP oszczędza wiele pracy, ponieważ większość niezbędnych konwersji przeprowadza za Ciebie. Jest to kolejne miejsce, gdzie PHP różni się całkowicie od większości języków programowania, które zwykle pozwalają na przeprowadzanie operacji jedynie na zmiennych o tych samych typach. W PHP możesz wykonać następującą operację:

```
$EngineType = "2.0L";
```

```
$TaxRate = 3;
$TaxPaid = $EngineType * $TaxRate;
```

Nie zmienia to zawartości żadnej ze zmiennych. Aby przekonać się, że wszystko jest w porządku, uruchom ten przykład.

Niejawne konwersje

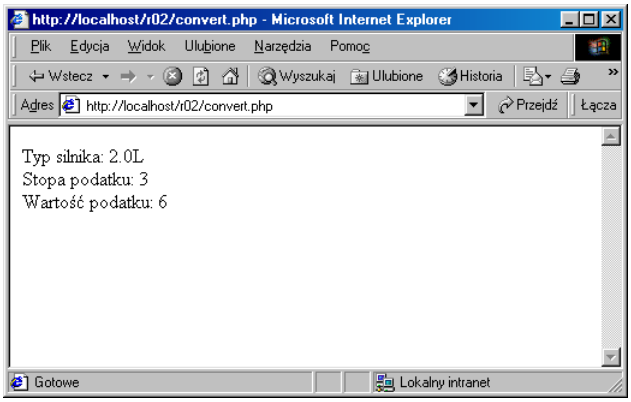
1. Otwórz edytor i wpisz następujący kod:

```
<HTML>
<BODY>
<?php
$EngineType = "2.0L";
$TaxRate = 3;
$TaxPaid = $EngineType * $TaxRate;
echo "Typ silnika: $EngineType<BR>";
echo "Stopa podatku: $TaxRate<BR>";
echo "Wartość podatku: $TaxPaid";
?>
</BODY>
</HTML>
```

2. Zapisz go *jako convert.php*.

3. Otwórz stronę w przeglądarce.

Rysunek 2.8.



Jak to działa?

Jest to przykład bardzo podobny do poprzedniego pokazującego operacje matematyczne. W pierwszych trzech liniach tworzone są trzy zmienne. Pierwsza z nich, `$EngineType` jest ciągiem:

```
$EngineType = "2.0L";
```

Druga zmienna, `$TaxRate` jest liczbą:

```
$TaxRate = 3;
```

W trzeciej linii przeprowadzamy nasze obliczenia, mnożąc zmienną `$EngineType` i `$TaxRate`, a wynik zapisując do zmiennej `$TaxPaid`:

```
$TaxPaid = $EngineType * $TaxRate;
```

W tym miejscu PHP przekazało informację: „Nie przejmuj się, że w ciągu przechowywanym w `$EngineType` jest litera L, widzę liczbę i tej liczby użyję w operacji mnożenia". Ostatnie trzy linie wyświetlają zawartość zmiennych, aby udowodnić, że zmienne nie uległy zmianie.

```
echo "Typ silnika: $EngineType<BR>";
echo "Stopa podatku: $TaxRate<BR>";
echo "Wartość podatku: $TaxPaid";
```

Rzutowanie typów

Do tej pory PHP sam przeprowadzał konwersje typów, ale gdy chcesz określić typ zmiennej w czasie jej tworzenia, również możesz to zrobić. Zrealizujesz to przez przypisanie do zmiennej własnej wartości i określenie żądanego typu poprzez podanie jego nazwy w nawiasach przed drugim wystąpieniem nazwy zmiennej. Operacja taka nazywana jest **rzutowaniem**.

```
$NowaZmienna = 13;
$NowaZmienna = (string) $NowaZmienna;
```

W naszym kodzie przypisujemy do zmiennej wartość numeryczną. Następnie w drugiej linii zamieniamy wartość numeryczną na ciąg. Jeżeli chcesz, możesz również skonwertować zmienną z powrotem na liczbę.

```
$NowaZmienna = 13;
$NowaZmienna = (string) $NowaZmienna;
$NowaZmienna = (integer) $NowaZmienna;
```

gettype i settype

Wspominaliśmy o funkcjach PHP pozwalających sprawdzić bieżący typ danych zmiennej. Funkcja taka nazywa się `gettype()`.

Jako argument funkcji podajemy nazwę zmiennej do sprawdzenia.

```
gettype($numer);
```

Aby wyświetlić typ zmiennej, przekażemy wynik działania tej funkcji do funkcji `echo()`.

```
$numer = 5;
echo gettype($numer);
```

Spowoduje to wyświetlenie na stronie WWW napisu `Integer`. Istnieje również analogiczna funkcja PHP o nazwie `settype()`, która podobnie do rzutowania typów

pozwala na określenie typu zmiennej. Wymaga ona dwóch argumentów: zmiennej, której typ ustawiamy i nazwy typu, na który zmieniamy zmienną. Działa ona następująco:

```
$numer = 10;
settype($numer, "string");
```

Aby pokazać, że typ zmiennej uległ zmianie, możesz wyświetlić na ekranie wartość zwracaną przez `gettype()`.

```
echo gettype($numer);
```

Wcześniej linia ta wyświetlała `Integer`, teraz wyświetla `String`.

isset, unset i empty

PHP posiada więcej funkcji używanych do obsługi zmiennych. Jedną z nich jest funkcja `isset()`, która pozwala sprawdzić, czy zmienna o podanej nazwie została utworzona. Jako argument przekazujemy nazwę sprawdzanej zmiennej. Jeżeli użyjesz tej funkcji jako argumentu funkcji `echo()`, to w przypadku gdy zmienna istnieje, wyświetlona zostanie liczba 1. W przeciwnym razie nie zostanie zwrócona żadna wartość, nawet zero.

```
echo isset($numer);
```

Drugą jest funkcja `unset()`, która usuwa zmienną i zwalnia pamięć przez nią używaną. Posiada również jeden argument, nazwę zmiennej:

```
unset($numer);
```

Przed jej użyciem upewnij się, czy wiesz co robisz, ponieważ funkcja ta usuwa zarówno nazwę zmiennej, jak i jej wartość.

Ostatnią z trzech przedstawianych funkcji jest `empty()`, logiczna odwrotność funkcji `isset()`. Używa się jej identycznie jak `isset()`. Zwraca 1, jeżeli zmienna nie istnieje lub ma wartość 0 albo "" (pusty ciąg). Funkcja nie zwraca żadnej wartości, jeżeli zmienna istnieje.

```
echo empty($numer);
```

Doszliśmy prawie do końca naszego wprowadzenia do zmiennych. Istnieje jeszcze jeden rodzaj zmiennych, o którym nie mieliśmy okazji opowiedzieć.

Zmienne środowiska

Zmienne środowiska (nazywane również zmiennymi PHP) to takie, których wartość ustawiana jest poza skryptem PHP, ale są w nim dostępne. Dostarczają zwykle danych na temat komunikacji klient-serwer, którą opisywaliśmy we wcześniejszej części tego rozdziału.

Mogą to być dane na temat żądania lub odpowiedzi HTTP. Mają taki sam format, jak zmienne tworzone przez Ciebie, zawierają na początku znak dolara. Jedyną różnicą jest to, że są utworzone przed rozpoczęciem wykonywania skryptu bez żadnej interwencji ze strony użytkownika. Możesz obejrzeć te zmienne w wyniku działania funkcji `phpinfo()`, użytej w pierwszym rozdziale.

Możesz również odwoływać się do nich za pomocą, na przykład funkcji `echo()`:

```
echo $HTTP_COOKIE_DATA;
```

Linia ta wyświetli dane wszystkich używanych cookie. Inną użyteczną zmienną środowiska jest `$HTTP_USER_AGENT`, która zawiera typ przeglądarki użytkownika.

```
echo $HTTP_USER_AGENT;
```

Możesz użyć jej w swoich programach w celu dostosowania wynikowych stron do specyficznej przeglądarki lub grupy przeglądarek. Kolejnymi użytecznymi zmiennymi są: `$HTTP_FROM`, która podaje adres e-mail użytkownika wysyłającego żądanie i `$HTTP_ACCEPT`, która zawiera listę różnych typów mediów, przyjmowanych przez przeglądarkę użytkownika.

Różnicą pomiędzy zmiennymi środowiska a stałymi zdefiniowanymi przez PHP jest to, że użytkownik może zmieniać wartość zmiennych. Jednak istnieją one raczej dla celów informacyjnych i nie powinno się ich zmieniać. W tej książce będziemy je tak właśnie traktować.

Podsumowanie

W tym rozdziale omówiliśmy sporo tematów podstawowych, więc aby umożliwić wszystkim zrozumienie materiału, staraliśmy się tłumaczyć poruszane zagadnienia możliwie jasno i zwięźle. Rozpoczęliśmy rozdział od wprowadzającego przykładu PHP pokazującego, że strony PHP są połączeniem tekstu, znaczników HTML i skryptów PHP. Skrypty PHP są wysyłane do serwera w celu przetworzenia.

W skrócie opisaliśmy proces interpretacji skryptów. Maszyna skryptowa PHP zwraca stronę składającą się z czystego HTML, więc przeglądarka nie ma kłopotów z jej zrozumieniem. Zapoznaliśmy się również z interakcją pomiędzy serwerem WWW a przeglądarką i sposobem przesyłania komunikatów HTTP.

Pokazaliśmy, w jaki sposób oznacza się skrypty PHP na stronie i krótko opisaliśmy proces buforowania stron. Rozpoczęliśmy programowanie wprowadzeniem zmiennych — metody, której PHP używa do przechowywaniu danych (tak jak większość języków programowania). Wymieniliśmy różne typy danych stosowanych do przechowywania danych i opisaliśmy dokładniej przechowywanie danych numerycznych i znakowych.

Zajęliśmy się procesem konwersji zmiennych pomiędzy typami oraz przeprowadziliśmy wprowadzenie do stałych, które są zmiennymi z niezmienną wartością. Zapoznaliśmy

się z niektórymi narzędziami PHP pozwalającymi wykonać podstawowe manipulacje tymi informacjami. Na koniec krótko opisaliśmy zmienne środowiska.