helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

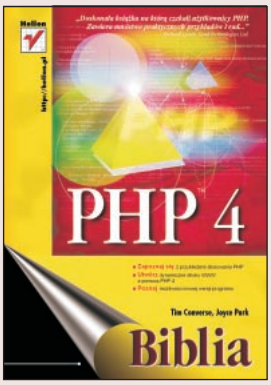
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

# PHP 4. Biblia




Autorzy: Tim Converse, Joyce Park
Tłumaczenie: Paweł Gonera
ISBN: 83-7197-391-8
Tytuł oryginału: PHP 4. Bible
Format: B5, stron: 608
oprawa twarda

Chcesz tworzyć dynamiczne strony WWW współpracujące z bazami danych? Ten rzetelnie opracowany podręcznik zawiera analizy konkretnych problemów. Ułatwi zapoznanie się z najnowszą wersją języka umożliwiającego tworzenie skryptów dołączanych do stron HTML. Jeżeli jesteś projektantem HTML, programistą C lub twórcą stron WWW używającym języków ASP, JSP, Perl lub ColdFusion, podręcznik ten stanie się twoim przewodnikiem po PHP – produkcie dostępnym bezpłatnie. Dzięki zawartym w książce wskazówkom zamiast rozpoczynać pracę „od zera", możesz wykorzystać powszechnie dostępne skrypty, zmieniając je odpowiednio do swoich potrzeb. Kompletny opis PHP 4 to między innymi rozbudowane rozdziały pomagające Ci wykorzystać szybkość relacyjnych baz danych oraz umiejętne łączenie stron WWW z bazami danych.
Naucz się, jak:

- tworzyć witryny ze śledzeniem sesji;
- używać PHP do programowania obiektowego;
- przyłączać kod PHP bezpośrednio do programów do obsługi poczty elektronicznej;
- zabezpieczać witrynę przed atakami;
- wykorzystać praktycznie mechanizmy cookie i przekierowywania.

Jeśli PHP 4 może coś wykonać, ty tym bardziej to potrafisz...

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

# Spis treści

**O Autorach** ..... **19**

**Przedmowa** ..... **21**

**Część I Podstawy PHP** ..... **27**

**Rozdział 1. Dlaczego PHP?** ..... **29**

Co to jest PHP? ..... 29
Historia PHP ..... 30
Dlaczego kochamy PHP? ..... 31
PHP jest darmowy ..... 31
PHP jest łatwy ..... 32
PHP można wbudować ..... 33
PHP jest niezależny ..... 35
PHP nie bazuje na znacznikach ..... 35
PHP jest stabilny ..... 36
PHP jest szybki ..... 36
PHP jest otwarty ..... 37
PHP dobrze współpracuje z innymi produktami ..... 38
Popularność PHP rośnie ..... 38
PHP nie jest niczyją własnością ..... 39
Społeczność PHP ..... 40
Podsumowanie ..... 40

**Rozdział 2. Skrypty wykonywane na serwerze WWW** ..... **41**

Statyczny HTML ..... 41
Technologie wykonywane na kliencie ..... 44
Skrypty wykonywane na serwerze ..... 47
Do czego są dobre skrypty serwera ..... 51
Podsumowanie ..... 53

**Rozdział 3. Rozpoczynamy pracę z PHP** ..... **55**

Dzierżawa lub własny serwer ..... 55
Wariant z dostawcą Internetu ..... 55
Własny serwer: wady i zalety ..... 58
Rozwiązania pośrednie ..... 59
Instalowanie PHP ..... 59
Zanim zaczniesz ..... 60
Procedura instalacji ..... 61
Narzędzia programistyczne ..... 66
Podsumowanie ..... 67

**Rozdział 4. Dodajemy PHP do HTML....69**
HTML jest gotowy na PHP....69
Przełączanie się z HTML do PHP....70
Kanoniczne znaczniki PHP....70
Krótkie znaczniki otwierające (w stylu SGML)....70
Witaj świecie....71
Wejście i wyjście z trybu PHP....72
Dołączanie plików....73
Podsumowanie....74

**Rozdział 5. Składnia, zmienne i wyświetlanie....75**
PHP wiele wybacza....75
HTML to nie PHP....76
Składnia PHP bazuje na C....76
PHP nie przejmuje się odstępami....76
PHP jest czasami wrażliwy na wielkość liter....77
Instrukcje to wyrażenia zakończone średnikiem....77
Bloki....80
Komentarze....80
Komentarze wielowierszowe w stylu C....81
Komentarze jednowierszowe: # i //....81
Zmienne....82
PHP skorzystał ze stylu zmiennych Perl....82
Deklarowanie zmiennych....82
Przypisywanie zmiennym wartości....82
Zmiana wartości zmiennych....83
Nieprzypisane zmienne....83
Możesz dowolnie zmieniać tryby pracy....85
Wyjście....86
Echo i print....86
Zmienne i ciągi....87
Podsumowanie....88

**Rozdział 6. Typy w PHP....91**
Pierwsza zasada: nie przejmuj się....91
Brak deklaracji typów zmiennych....91
Automatyczna konwersja typów....92
Typy w PHP....92
Typy proste....93
Integer....93
Double....94
Boolean....95
Przykłady....96
String....97
Tablice....100
Implementacja tablic....101
Ciągi znaków jako indeksy tablicy....101
Czy w PHP są struktury?....102
Inne własności tablic....102
Obiekty....102
Przegląd OOP....102
Jak bardzo obiektowy jest PHP?....103
Definiowanie klas w PHP....103
Tworzenie obiektów....104

Kontrola typów ....104
Przypisania i konwersje ....105
Przepełnienie liczby całkowitej ....109
Szukamy największej liczby całkowitej ....109
Podsumowanie ....110

**Rozdział 7. Sterowanie ....111**
Wyrażenia logiczne ....112
Stałe logiczne ....112
Operatory logiczne ....112
Operatory porównania ....114
Operator trójskładnikowy ....116
Instrukcje warunkowe ....117
If-else ....117
Switch ....120
Pętle ....121
Pętle ograniczone i nieograniczone ....122
While ....122
Do-while ....123
For ....123
Przykłady pętli ....124
Break i continue ....126
Pętle nieskończone ....128
Składnia alternatywna ....129
Przerywanie wykonania ....129
Podsumowanie ....130

**Rozdział 8. Użycie i definiowanie funkcji ....133**
Użycie funkcji ....133
Zwracane wartości a efekty uboczne ....134
Dokumentacja funkcji ....134
Nagłówki w dokumentacji ....135
Szukanie opisu funkcji ....136
Definiowanie własnych funkcji ....136
Czym jest funkcja? ....136
Składnia definicji funkcji ....137
Przykład definicji funkcji ....137
Parametry formalne i parametry aktualne ....139
Nieprawidłowa liczba argumentów ....139
Funkcje a zasięg zmiennych ....139
Zmienne globalne i lokalne ....140
Zmienne statyczne ....141
Zasięg funkcji ....142
Include oraz require ....142
Rekurencja ....143
Zagadnienia zaawansowane ....144
Zmienna liczba argumentów ....145
Wywołanie przez wartość a wywołanie przez referencję ....148
Wywołanie przez referencję ....148
Zmienne jako nazwy funkcji ....150
Bardziej skomplikowany przykład ....150
Podsumowanie ....153

**Rozdział 9. Ciągi i funkcje operujące na ciągach ....155**
Ciągi w PHP ....155
Znaki i indeksy ciągu ....156
Operatory dla ciągów ....156
Złączenie i przypisanie ....157
Funkcje operujące na ciągach ....157
Sprawdzanie ciągów ....157
Szukanie znaków i podciągów ....158
Porównywanie i przeszukiwanie ....159
Przeszukiwanie ....160
Wycinanie podciągu ....161
Funkcje porządkujące ....163
Zastępowanie ciągów ....163
Ciągi i kolekcje znaków ....165
Funkcje analizujące ....167
Funkcje zmiany wielkości liter ....169
Funkcje znaków sterujących ....170
Formatowanie danych ....171
Zaawansowane własności ciągów ....173
Wyrażenia regularne ....173
Funkcje HTML ....176
Podsumowanie ....176

**Rozdział 10. Matematyka ....177**
Typy numeryczne ....177
Operatory matematyczne ....178
Operatory arytmetyczne ....178
Operatory arytmetyczne i typy ....178
Operator inkrementacji ....179
Operator przypisania ....180
Operatory porównania ....180
Kolejność operacji i nawiasy ....181
Proste funkcje matematyczne ....182
Konwersja podstawy ....184
Funkcje wykładnicze i logarytmy ....186
Trygonometria ....186
Liczby losowe ....190
Inicjowanie generatora ....190
Przykład: losowy wybór ....192
Arytmetyka o dowolnej dokładności ....193
Przykład użycia funkcji o dowolnej dokładności ....194
Konwersja obliczeń na dowolną dokładność ....195
Podsumowanie ....197

**Rozdział 11. Tablice i funkcje operujące na tablicach ....199**
Użycie tablic ....199
Czym są tablice PHP? ....200
Tworzenie tablic ....202
Bezpośrednie przypisanie ....202
Konstrukcja array() ....203
Podawanie indeksów przy użyciu array() ....203
Funkcje zwracające tablice ....204
Odczytywanie wartości ....204
Konstrukcja list() ....205

Tablice wielowymiarowe ... 206
Informacje o tablicach ... 207
Usuwanie z tablicy ... 207
Iteracje ... 208
Użycie funkcji iteracyjnych ... 208
Iteracje za pomocą current() i next() ... 210
Powtórne przeglądanie za pomocą reset() ... 211
Wypisywanie w odwrotnym porządku za pomocą end() i prev() ... 212
Pobieranie wartości kluczy za pomocą key() ... 213
Wartości puste i funkcja each() ... 213
Przeglądanie tablicy za pomocą array_walk() ... 214
Stosy i kolejki ... 215
Przekształcenia tablic ... 218
Pobieranie kluczy i wartości ... 218
Zamiana, odwracanie i mieszanie ... 219
Zamiana pomiędzy tablicą i zmiennymi ... 222
Sortowanie ... 222
Podsumowanie ... 223

**Rozdział 12. Przekazywanie danych pomiędzy stronami ... 225**
HTTP jest protokołem bezstanowym ... 225
Argumenty GET ... 226
Inne zastosowania adresów URL w stylu GET ... 228
Argumenty POST ... 230
Zarządzanie zmiennymi w PHP ... 232
Podsumowanie ... 234

**Rozdział 13. Funkcje systemu operacyjnego i dostępu do plików ... 235**
Funkcje czytania i zapisywania plików ... 236
Otwarcie pliku ... 236
Czytanie pliku ... 238
Zapis do pliku ... 239
Zamknięcie pliku ... 241
Funkcje systemu plików i katalogów ... 241
feof ... 241
file_exists ... 241
filesize ... 244
Funkcje sieciowe ... 244
Funkcje logu systemowego ... 244
Funkcje DNS ... 244
Funkcje gniazd ... 245
Funkcje daty i czasu ... 245
Jeżeli nie znasz daty ani czasu ... 246
Jeżeli już odczytałeś datę i czas ... 247
Funkcje konwersji kalendarza ... 247
Podsumowanie ... 249

**Rozdział 14. Styl PHP ... 251**
Zalety prawidłowego stylu ... 251
Czytelność ... 252
Komentarze ... 256
Nazwy zmiennych i plików ... 257
Łatwość konserwacji ... 259
Unikaj „magicznych liczb” ... 259
Funkcje ... 260

Pliki dołączane ....260
Interfejs obiektowy ....262
Solidność ....263
Niedostępność usługi ....263
Niespodziewany typ zmiennej ....263
Zwięzłość i wydajność ....264
Używaj właściwych algorytmów ....264
Poprawianie wydajności ....264
Zwięzłość: zmniejszanie ....265
Wskazówki na temat zwięzłości ....266
Tryb HTML, czy PHP? ....268
Oddzielanie kodu od projektu ....274
Funkcje ....274
Arkusze stylów w PHP ....274
Szablony i spójność stron ....275
Podsumowanie ....276

**Rozdział 15. Podstawowe pułapki PHP ....277**
Problemy związane z instalacją ....277
Źródło pliku wyświetlane w przeglądarce ....278
Blok PHP pokazuje się jako tekst; przeglądarka chce zapisać plik ....278
Nieodnaleziony serwer lub niemożliwe wyświetlenie strony ....278
Problemy z wyświetlaniem ....279
Całkowicie pusta strona ....279
Niekompletna lub nieprawidłowa strona ....279
Kod PHP pokazuje się w przeglądarce ....281
Niepowodzenie przy ładowaniu strony ....282
Nieodnaleziona strona ....282
Nieudane otwarcie pliku do włączenia ....283
Błędy analizy składni ....283
Komunikat błędu składni ....283
Brakujący średnik ....284
Brak znaku $ ....284
Nieprawidłowa zmiana trybu ....285
Nieoznaczone apostrofy ....285
Inne błędy składni ....286
Uprawnienia do plików ....286
Błąd HTTP nr 403 ....286
Brak dołączanych plików ....286
Ostrzeżenie przy włączaniu pliku ....287
Nieprzypisane zmienne ....287
Zmienna nie pokazuje się w wynikowym ciągu ....287
Jak zachowują się niezainicjowane zmienne ....288
Problemy z wielkością liter ....288
Problemy z zasięgiem ....288
Problemy z funkcjami ....289
Wywołanie niezdefiniowanej funkcji ....289
Nie można ponownie zadeklarować funkcji ....290
Nieprawidłowa liczba argumentów ....290
Błędy matematyczne ....290
Ostrzeżenie o dzieleniu przez zero ....290
Niespodziewane wyniki działań ....291
NaN (lub NAN) ....291

Przekroczenie czasu oczekiwania ....292
Podsumowanie ....292

Część II PHP i bazy danych ....297

Rozdział 16. Wybór bazy danych dla PHP ....299
Czemu używamy baz danych? ....299
Unikanie redundancji ....300
Unikanie nudnego programowania ....300
Szukanie ....300
Bezpieczeństwo ....301
Architektura wielowarstwowa ....301
Wybór bazy danych ....302
Możesz nie mieć wyboru ....302
Plikowe, relacyjne i obiektowo-relacyjne bazy danych ....302
ODBC/JDBC kontra własne API ....303
Zmiana bazy danych ....304
Przegląd zaawansowanych funkcji ....304
GUI ....304
Podzapytania ....304
Złożone złączenia ....305
Wielowątkowość i blokowanie ....305
Transakcje ....305
Procedury i wyzwalacze ....306
Klucze obce i więzy integralności ....306
Replikacja bazy danych ....306
Bazy danych obsługiwane przez PHP ....307
Wybieramy MySQL ....307
Podsumowanie ....308

Rozdział 17. Samouczek SQL ....311
Standardy SQL ....311
Podstawowe wyrażenia SQL ....312
SELECT ....312
INSERT ....315
UPDATE ....316
DELETE ....316
Projekt bazy danych ....316
Użycie połączeń do bazy danych ....319
Bezpieczeństwo i uprawnienia ....319
Ustawianie uprawnień ....320
Przechowywanie haseł w innym miejscu ....320
Użycie formularzy PHP do sprawdzania haseł ....321
Tworzenie kopii bezpieczeństwa ....322
Podsumowanie ....322

Rozdział 18. Funkcje PHP i MySQL ....325
Łączenie z MySQL ....325
Tworzenie zapytań w MySQL ....326
Pobieranie wyniku ....327
Pobieranie opisu danych ....330
Korzystanie z wielokrotnych połączeń ....330
Kontrola błędów ....332
Tworzenie baz danych MySQL za pomocą PHP ....332

Funkcje MySQL....333
Podsumowanie....335

**Rozdział 19. Wyświetlanie zapytań w tabelach....337**
Tabele HTML i tabele bazy danych....338
Przekształcenie jeden w jeden....338
Przykład: wyświetlanie jednej tabeli....338
Przykładowe tabele....340
Ulepszanie wyświetlania....341
Złożone odwzorowania....343
Wiele zapytań albo skomplikowane wyświetlanie....344
Użycie kilku zapytań....345
Przykład skomplikowanego wyświetlania....346
Tworzenie przykładowych tabel....348
Podsumowanie....350

**Rozdział 20. Tworzenie formularzy z zapytań....351**
Formularze HTML....351
Samoprzetwarzanie....352
Obsługa formularzy....353
Formularze zależne od zmiennych....356
TEXT i TEXTAREA....356
CHECKBOX....358
RADIO....359
SELECT....359
Formularze zależne od zapytań....361
Podsumowanie....362

**Rozdział 21. Dziennik sieciowy....363**
Dlaczego dziennik?....363
Najprostszy dziennik....364
Wprowadzanie danych przez HTTP....368
Dołączenie bazy danych....370
Możliwe rozszerzenia....375
Podsumowanie....376

**Rozdział 22. Sieciowe głosowanie....377**
Zadania systemu....377
Cele systemu....378
Struktura....378
Obsługa bazy danych....379
Zbieranie głosów....379
Wyświetlanie sumarycznych wyników....383
Nadużycia i skalowanie....387
Podsumowanie....387

**Rozdział 23. Styl i efektywność rozwiązań na podstawie PHP i bazy danych....389**
Połączenia — ograniczanie i powtórne użycie....390
Przykład nieprawidłowego użycia: jedno połączenie na wyrażenie....390
Kilka wyników nie wymaga kilku połączeń....391
Trwałe połączenia....391
Przenoszenie pracy na serwer bazy danych....392
Baza jest szybsza od ciebie....392
Przykład nieprawidłowego użycia: pętla zamiast warunku....393

Tworzenie pól daty i czasu ....394
Szukanie ostatnio wstawionego wiersza ....395
Podsumowanie ....397

**Rozdział 24. Pułapki tandemu PHP-baza danych ....399**
Brak połączenia ....399
Problemy z uprawnieniami ....402
Nieoznaczone apostrofy ....403
Nieprawidłowe zapytania SQL ....405
Pomyłki w nazwach ....407
Pomyłki przy przecinkach ....407
Ciągi nieotoczone apostrofami ....407
Niezainicjowane zmienne ....407
Zbyt mało danych, zbyt dużo danych ....408
Kontrola poprawności ....409
Podsumowanie ....409

**Część III Techniki zaawansowane ....411**

**Rozdział 25. Sesje ....413**
Czym są sesje? ....413
Co stanowi problem? ....413
Dlaczego się tym zajmujemy? ....414
Alternatywy sesji ....414
Adres IP ....414
Ukryte zmienne ....415
Cookie ....416
Jak działają sesje w PHP ....416
Uaktywnianie sesji w PHP ....417
Rejestrowanie zmiennych w sesji ....418
Gdzie są przechowywane dane? ....419
Funkcje obsługi sesji ....420
Przykładowy kod sesji ....422
Zagadnienia konfiguracji ....423
Pułapki i wykrywanie usterek ....423
Podsumowanie ....426

**Rozdział 26. Cookie i HTTP ....427**
Cookie ....427
Funkcja setcookie() ....428
Przykłady ....428
Usuwanie cookie ....430
Odczytywanie cookie ....431
Zmienne GET, POST i cookie ....432
Pułapki cookie ....435
Wysyłanie nagłówków HTTP ....437
Przykład: przekierowanie ....437
Przykład: uwierzytelnianie HTTP ....438
Pułapki związane z nagłówkami ....439
Podsumowanie ....439

**Rozdział 27. PHP i JavaScript ....441**
Tworzenie kodu JavaScript w PHP ....441
Pojedynek obiektów ....442
PHP nie analizuje wysyłanych danych ....442
Kiedy używać JavaScript ....444

PHP jako koło zapasowe do JavaScript ....444
JavaScript statyczny kontra dynamiczny ....445
Dynamiczna generacja formularzy ....446
Przesyłanie danych z JavaScript do PHP ....450
Podsumowanie ....452

**Rozdział 28. E-mail ....455**
Informacje na temat architektury e-mail ....455
Model systemu e-mail ....456
Pobieranie poczty za pomocą PHP ....460
Tworzenie przez zaniechanie ....461
Tworzenie przez przykład ....461
Tworzenie przez upiększanie ....461
Wysyłanie poczty za pomocą PHP ....462
Konfiguracja Windows ....462
Konfiguracja Unixa ....462
Funkcja mail ....462
Więcej na temat aplikacji pocztowych ....464
Wysyłanie poczty z formularza ....464
Wysyłanie poczty przy użyciu bazy danych ....466
Własna aplikacja pocztowa w PHP ....466
Podsumowanie ....468

**Rozdział 29. PHP i XML ....469**
Co to jest XML? ....469
Praca z XML ....472
Dokumenty i DTD ....472
Struktura DTD ....474
Analizatory kontrolujące i nie kontrolujące poprawności ....476
DOM kontra SAX ....477
SAX ....477
DOM ....478
Funkcje PHP dla DOM ....478
SAX ....480
Użycie SAX ....481
Opcje SAX ....482
Funkcje PHP dla SAX ....483
Przykładowa aplikacja SAX ....486
Pułapki i wyszukiwanie błędów ....491
Podsumowanie ....492

**Rozdział 30. Programowanie obiektowe w PHP ....493**
Jak dobre jest programowanie obiektowe? ....494
Terminologia programowania obiektowego ....494
Obiekty, klasy i typy w PHP ....495
Atrybuty ....496
Funkcje ....496
Konstruktory ....497
Dziedziczenie ....497
Przesłanianie ....498
Przeciążanie ....499
Zasięg ....499
Przypisywanie, aliasy i referencje ....500
Wyświetlanie i drukowanie obiektów ....502
Przeglądanie ....503

Funkcje przeglądania typów i klas ... 503
Serializacja obiektów ... 507
Zewnętrzne interfejsy: COM, Java i CORBA ... 508
COM i DCOM ... 509
Przykładowa aplikacja obiektowa ... 510
Podsumowanie ... 513

**Rozdział 31. Bezpieczeństwo i kryptografia ... 515**
Możliwe ataki ... 516
Zmiana zawartości witryny ... 516
Dostęp do kodu źródłowego ... 518
Odczyt dowolnego pliku ... 519
Uruchamianie dowolnych programów ... 521
Wirusy i inne e-robaki ... 523
Bezpieczeństwo poczty ... 523
Szyfrowanie ... 524
Szyfrowanie kluczem publicznym ... 524
Szyfrowanie pojedynczym kluczem ... 526
Szyfrowanie cookie ... 527
Mieszanie ... 529
Cyfrowe podpisywanie plików ... 530
Secure Server Layer ... 531
Witryny podejmujące problematykę bezpieczeństwa ... 531
Podsumowanie ... 532

**Rozdział 32. Konfiguracja i dostrajanie ... 535**
Podglądanie zmiennych środowiska ... 535
Poznajemy konfigurację PHP ... 535
Opcje kompilacji ... 536
Opcje kompilacji dla postaci CGI ... 541
Pliki konfiguracyjne Apache ... 543
Plik php.ini ... 545
Poprawianie wydajności PHP ... 550
Podsumowanie ... 552

**Dodatki ... 553**

**Dodatek A PHP dla programistów C ... 555**

**Dodatek B PHP dla programistów ASP ... 561**

**Dodatek C PHP dla programistów HTML ... 571**

**Dodatek D Zasoby Sieci na temat PHP ... 579**

**Słownik ... 587**

**Skorowidz ... 597**

Rozdział 12.

# Przekazywanie danych pomiędzy stronami

W tym rozdziale:

- Dlaczego HTTP jest jak toczący się kamień?
- Argumenty GET
- Inne użycie adresów URL w stylu GET
- Argumenty POST
- Obsługa zmiennych GET/POST w PHP

W tym rozdziale zajmiemy się przekazywaniem danych, na przykład zmiennych, pomiędzy stronami WWW. Niektóre z tych informacji nie należą jedynie do PHP, ale są konsekwencją interakcji PHP i HTML lub samego protokołu HTTP.

## HTTP jest protokołem bezstanowym

Ważne, abyś pamiętał, że sam protokół HTTP jest protokołem bezstanowym. Oznacza to, że każde wywołanie HTTP jest niezależne od innych, nie ma informacji na temat stanu klienta i nie jest zapamiętywane. Każde wywołanie powoduje utworzenie osobnego wątku, który ma za zadanie dostarczyć jeden plik, następnie jest niszczony (uściślając, wraca do puli dostępnych wątków).

Nawet jeżeli projekt twojej witryny zakłada nawigację tylko w jedną stronę (ze strony 1. tylko do strony 2., a następnie do strony 3. itd.), protokół HTTP nigdy nie będzie wiedział, czy użytkownik oglądający stronę 2. załadował wcześniej stronę 1. Nie możesz ustawić na stronie 1. zmiennej i oczekiwać, że zostanie ona zaimportowana do strony 2. tylko przez mechanizmy HTML. Możesz użyć HTML do wyświetlenia formularza, do którego można wprowadzić informacje, ale jeżeli nie zrealizujesz mechanizmów przekazujących dane do innej strony lub programu, zmienne znikną w momencie przejścia na inną stronę.

Dla takich przypadków stworzone zostały technologie przetwarzania formularzy, np. PHP. PHP przechwytuje zmienne przenoszone z jednej strony do drugiej i pozwala na ich późniejsze wykorzystanie. Jest niezwykle użyteczny do funkcji przenoszenia danych i pozwala na ich łatwe zastosowanie w wielu przypadkach.

Istnieją bardziej zaawansowane sposoby zapamiętania interakcji klienta z witryną, jak *cookie* oraz sesje. W tym rozdziale skupimy się tylko na podstawowych technikach przenoszenia danych pomiędzy stronami z zastosowaniem metod `GET` i `POST` z HTML.



Programiści ASP niezmiennie twierdzą, że PHP jest gorszy, ponieważ uważają, że zmienne sesji są czymś niezwykłym. Nie daj się zakrzyczeć, Microsoft używa mechanizmu *cookie* do przechowywania zmiennych sesji, co może powodować wiele problemów. Możesz o tym przeczytać w dodatku B.

# Argumenty GET

Metoda `GET` pozwala na przesyłanie argumentów pomiędzy stronami za pomocą fragmentu adresu URL. W przypadku użycia tej metody do obsługi formularzu, `GET` powoduje dodanie nazw zmiennych i ich wartości do adresu URL podanego jako atrybut `ACTION`. Poszczególne zmienne rozdzielone są znakiem zapytania. Kompletny adres URL przesyłany jest do przetworzenia (w tym przypadku przez PHP).

Poniżej przedstawiamy przykładowy formularz HTML używający metody `GET`.

```
<HTML>
<HEAD>
<TITLE>Przykład GET część 1</TITLE>
</HEAD>
<BODY>
<FORM ACTION="http://localhost/php/baseball.php" METHOD="GET">
<P>Root, root, root for the:<BR>
<SELECT NAME="Team" SIZE=2>
<!-- Dobrze jest używać atrybutu VALUE, nawet w przypadkach, gdy
nie jest obowiązkowy. W tym przykładzie jest on absolutnie niezbędny.
-->
<OPTION VALUE="Cubbies">Chicago Cubs (National League)
<OPTION VALUE="Pale Hose">Chicago White Sox (American League)
</SELECT>
<P><INPUT TYPE="submit">
</FORM>
</BODY>
</HTML>
```

Gdy użytkownik wybierze jedną z wartości i kliknie przycisk, przeglądarka sklei bez dodatkowych odstępów następujące elementy:

- adres URL podany w apostrofach po słowie `ACTION` (*http://localhost/php/baseball.php*);
- znak zapytania (?);

- nazwę zmiennej, znak równości i jej wartość (Team=2);
- znak & oraz kolejną parę `NAME=VALUE` (Submit=Submit). Ta część może być powtarzana tyle razy, na ile pozwala ograniczenie długości adresu przez serwer.

Dla naszego przykładu utworzony zostanie następujący ciąg URL:

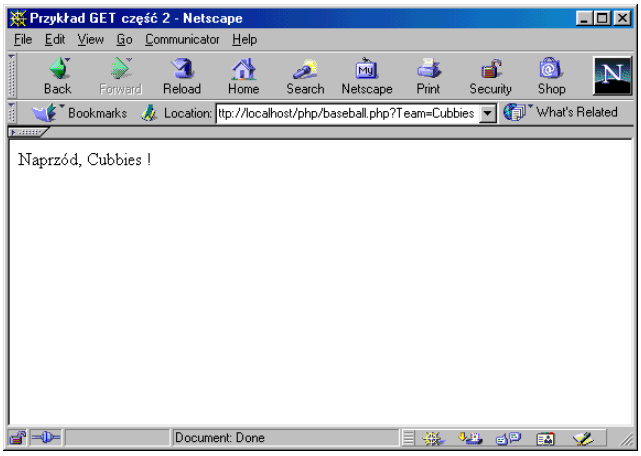
```
http://localhost/php/baseball.php?Team=Cubbies&Submit=Submit
```

Adres ten jest przesyłany przez przeglądarkę jako nowe żądanie. Skrypt PHP, do którego zostało przesłane powyższe żądanie, pobiera z adresu wartości `GET` i używa ich do stworzenia strony. W naszym przypadku wkleimy wybraną wartość do tekstu.

```
<HTML>
<HEAD>
<TITLE>Przykład GET część 2</TITLE>
<STYLE TYPE="text/css">
<!--
BODY {font-size: 24pt;}
-->
</STYLE>
</HEAD>
<BODY)
<P>Naprzód,
<?php print("$Team"); ?> !
</BODY>
</HTML>
```

Końcowy wynik przedstawiony jest na rysunku 12.1.

**Rysunek 12.1.** *Wynik działania formularza używającego METHOD=GET*




Obsługa formularzy za pomocą metody `GET` posiada jedną wielką zaletę w porównaniu z metodą `POST`. Tworzy ona różne adresy, które użytkownik może zapamiętać. Z wyniku działania formularza bazującego na metodzie `POST` nie da się zrobić zakładki.

Sposób obsługi formularzy za pomocą metody `GET` posiada tyle wad, że standard HTML 4.0 odradza jego stosowanie. Wadami tymi są między innymi:

- `GET` nie jest odpowiedni do logowania użytkownika, ponieważ nazwa użytkownika i hasło są widoczne na ekranie i mogą być zapisane w pamięci odwiedzonych stron w przeglądarce.

- Wysłane przez formularz dane zapamiętywane są w logach serwera WWW.
- Ponieważ `GET` przypisuje dane do zmiennej środowiskowej serwera, jej długość jest ograniczona. Próba przesłania np. 300 słów w adresie URL nie jest najlepszym pomysłem.



Oryginalna specyfikacja HTML określa maksymalną długość żądań na 244 znaki. Mimo że ograniczenie to zostało później zarzucone, używanie dłuższych ciągów nie jest dobrym pomysłem.

Ponieważ można zapamiętywać adresy utworzone przez metodę `GET`, pojawiło się wiele głosów za zachowaniem tej metody przez W3. Tak się stało i mimo że jest to metoda najczęściej używana do obsługi formularzy, jest zalecana tylko dla powtarzalnych zastosowań, czyli takich, które nie powodują stałego efektu ubocznego. Najbardziej odpowiednim zastosowaniem `GET` jest okno przeszukiwania. Dla innych formularzy lepiej użyć metody `POST`.

# Inne zastosowania adresów URL w stylu GET

Mimo że w chwili obecnej obsługiwanie formularzy przez metodę `GET` nie jest zalecane, adresy URL w tym stylu są bardzo użyteczne do obsługi nawigacji po witrynie. Szczególnie użyteczne jest użycie ich w witrynach generowanych dynamicznie, które są często tworzone przy użyciu PHP, ponieważ adresy zawierające dołączone zmienne świetnie współpracują z systemem szablonów zależnych od kontekstu.

Załóżmy, że jesteś właścicielem witryny traktującej o zwierzętach hodowanych dla wełny. Po długiej i ciężkiej pracy nad stroną informacyjną i graficzną masz następujące strony:

```
alpaca.html
guanaco.html
llama.html
vicuna.html
```

Lecz gdy witryna zacznie się rozrastać, płaska struktura plików może wymagać dużo pracy przy administracji, szczególnie gdy proste poprawki trzeba będzie wykonać na każdej stronie. Jeżeli struktura stron jest podobna, możesz zastanowić się nad zastosowaniem systemu szablonów razem z PHP.

Można zastosować pojedynczy szablon oraz oddzielne pliki tekstowe dla każdego zwierzęcia (zawierające dane, fotografie itp.):

```
fleecee.php
  alpaca.inc
  guanaco.inc
  llama.inc
  vicuna.inc
```

Można również użyć większej liczby wyspecjalizowanych szablonów:

```
goat.php
  cashmere.inc
insect.php
  silkworm.inc
llamoid.php
  alpaca.inc
rabbit.php
  angora.inc
sheep.php
  merino.inc
```

W obu przypadkach plik szablonu będzie wyglądał podobnie (nie zamieściliśmy wszystkich niezbędnych plików, więc przykład ten nie będzie działał):

```
<HTML>
<HEAD>
<TITLE>Zwierzęta hodowane dla wełny</TITLE>
<STYLE TYPE="text/css">
<!--
BODY {font: verdana; font-size: l4pt}
-->
</STYLE>
</HEAD>
<BODY>
<TABLE BORDER=0 CELLPADDING=0 WIDTH=100%>
<TR>
<!- Panel nawigacji z adresami w stylu GET. -->
<TD BGCOLOR="#428284" ALlGN=CENTER VALIGN=TOP WIDTH=25%>
<P>
<A HREF="fleecee.php?Name=alpaca"><B>Alpaca</B></A>
<BR>
<A HREF="fleecee.php?Name=guanaco"><B>Guanaco</B></A>
<BR>
<A HREF="fleecee.php?Name=llama"><B>Llama</B></A>
<BR>
<A HREF="fleecee.php?Name=vicuna"><B>Vicu&#241a</B></A>
<BR><BR>
</TD>
<TD BGCOLOR="#FFFFFF" ALIGN=LEFT VALIGN=TOP WIDTH=75%>
<? include("$Name.inc"); ?>
</TD></TR></TABLE>
</BODY>
</HTML>
```

Łącza na panelu nawigacji są obsługiwane przez przeglądarkę tak samo jak wynik działania metody `GET`.

Jednak nawet w przypadku takiego rozwiązania masz jeszcze wiele ręcznej pracy: należy się upewnić, że każdy włączany plik jest prawidłowo sformatowany, dodanie nowego pliku do witryny wymaga dodania nowego łącza i innych tego typu prac. Zgodnie z zasadą oddzielania formy od treści możesz przejść na wyższy poziom abstrakcji przy użyciu bazy danych. W tym przypadku adres typu:

```
fleecee.php?animalID=2
```

powinien spowodować sięgnięcie przez szablon do bazy danych (użycie zmiennej numerycznej zamiast słowa przyspiesza interakcję z bazą danych). Taki system powinien automatycznie umieszczać na panelu na podstawie zawartości bazy danych. Dzięki temu możesz tworzyć kolejne strony WWW bez interwencji.

# Argumenty POST

POST jest zalecaną metodą obsługi formularzy, szczególnie dla niepowtarzalnych zastosowań (takich, które skutkują trwałymi efektami działania), takich jak dodawanie danych do bazy. Zbiór danych formularza jest zawarty w ciele formularza w czasie jego wysyłania do agenta przetwarzania (w tym przypadku PHP). Nie ma żadnych widocznych zmian w adresie URL w zależności od przesyłanych danych.

Metoda POST posiada następujące zalety:

- Jest dużo bezpieczniejsza od GET, ponieważ wprowadzone przez użytkownika dane nie są widoczne w adresie URL, logach serwera oraz na ekranie (jeżeli są prawidłowo obsłużone).
- Maksymalny rozmiar przesyłanych danych jest dużo większy (kilka kilobajtów zamiast kilku setek znaków).

Metoda ta posiada również wady:

- Nie można utworzyć zakładki z wyniku działania.
- Metoda ta może być niezgodna z niektórymi ustawieniami firewalla, jeżeli z powodów bezpieczeństwa usuwa on dane z formularza.

W książce tej konsekwentnie używamy POST do obsługi formularzy umieszczających dane w systemie. GET używamy do nawigacji po witrynie oraz do obsługi przeszukiwania, czyli zastosowań wymagających odczytania danych i ich wyświetlenia.

Wiele, jeżeli nie większość zastosowań POST powoduje zainicjowanie połączenia w usługowej warstwie systemu. Przykład pokazany na wydruku 12.1 jednak tego nie wykonuje (ponieważ skupiamy się na metodzie POST, usunęliśmy sprawdzanie poprawności danych, więc formularz ten wymaga jeszcze pracy).

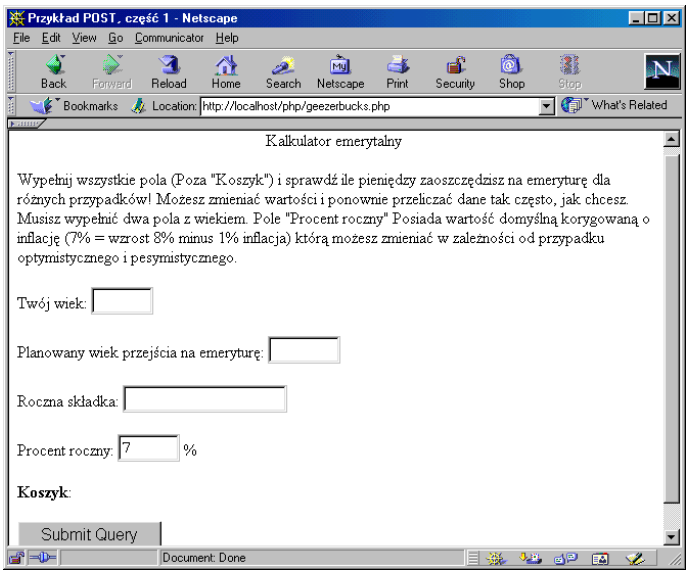
**Wydruk 12.1.** *Arkusz oszczędności emerytalnych*

```
<HTML>
<HEAD>
<TITLE>Przykład POST, część 1</TITLE>
<STYLE TYPE="text/css">
<!--
BODY        {font-size: 14pt}
.heading    {font-size: 18pt; color: red}
-->
</STYLE>
</HEAD>
<?php
/* To sprawdzenie pozwala na stwierdzenie, czy formularz
jest wyświetlony po raz pierwszy (w tym przypadku wyświetla tylko
domyślny procent roczny), czy już zostały wysłane wprowadzone wartości
 */
if (!IsSet($stage))
  $AnnGain = 7;
else
{
  $Years = $RetireAge - $CurrentAge;
  $YearCount = 0;
```

```
  $Total = $Contrib;
  while($YearCount <= $Years)
  {
    $Total = round($Total * (1.0 + $AnnGain/100) + $Contrib);
    $YearCount = $YearCount + 1;
  }
}
?>
<BODY>
<DIV ALIGN=CENTER ID=Div1 class=heading>
Kalkulator emerytalny
</DIV>
<P class=blurb>Wypełnij wszystkie pola (Poza "Koszyk") i sprawdź,
ile pieniędzy zaoszczędzisz na emeryturę dla różnych przypadków!
Możesz zmieniać wartości i ponownie przeliczać dane tak często,
jak chcesz. Musisz wypełnić dwa pola wieku. Pole "Procent roczny"
posiada wartość domyślną korygowaną o inflację
(7% = wzrost 8% minus 1% inflacja),
którą możesz zmieniać w zależności od przypadku optymistycznego i
pesymistycznego.</P>
<FORM ACTION="<?php print("$PHP_SELF"); ?>" METHOD="POST">
<P>Twój wiek: <INPUT TYPE="text" SIZE=5 NAME="CurrentAge"
VALUE="<?php print("$CurrentAge"); ?>">
<P>Planowany wiek przejścia na emeryturę: <INPUT TYPE="text"
SIZE=6 NAME="RetireAge" VALUE="<?php print("$RetireAge"); ?>">
<P>Roczna składka: <INPUT TYPE="text" SIZE=15
NAME="Contrib" VALUE="<?php print("$Contrib"); ?>">
<P>Procent roczny: <INPUT TYPE="text" SIZE=5 NAME="AnnGain"
VALUE="<?php print("$AnnGain"); ?>"> %
<BR><BR>
<P><B>Koszyk</B>: <?php print("$Total"); ?>
<P><INPUT TYPE="hidden" NAME="stage" VALUE=1>
<P><INPUT TYPE="submit">
</FORM>
</BODY>
</HTML>
```

Na rysunku 12.2 przedstawiony jest formularz z wydruku 12.1.

**Rysunek 12.2.** *Formularz używający metody POST*



# Zarządzanie zmiennymi w PHP

PHP tak efektywnie przenosi dane, ponieważ projektanci podjęli bardzo wygodną, lecz (w teorii) ryzykowną decyzję. PHP automatycznie i niewidocznie dla użytkownika przypisuje zmienne na nowej stronie, gdy wyślesz je za pomocą `GET` lub `POST`. Większość z jego konkurentów wymaga jawnego wykonania tych przypisań na każdej stronie. Jeżeli o tym zapomnisz lub popełnisz błąd, dane nie będą dostępne dla agenta przetwarzania. PHP jest szybszy, prostszy i w większości przypadków odporny na pomyłki.

Najprostszą metodą zilustrowania tej różnicy jest pokazanie różnych metod przetwarzania danych z tego samego formularza. Formularz ten wygląda następująco:

```
<HTML>
<HEAD>
<TITLE>Formularz preferencji słodyczy</TITLE>
</HEAD>
<BODY>
<FORM ACTION="candy.php" METHOD="POST">
Jakie słodycze lubisz najbardziej?<BR>
<INPUT TYPE="radio" NAME="Candy" VALUE="orzeszki ziemne">Orzeszki ziemne<BR>
<INPUT TYPE="radio" NAME="Candy" VALUE="snickers">Snickers<BR>
<INPUT TYPE="radio" NAME="Candy" VALUE="krówki">Krówki<BR>
<INPUT TYPE="submit">
</FORM>
</BODY>
</HTML>
```

Skrypt PHP obsługujący formularz:

```
<HTML>
<HEAD>
<TITLE>Odpowiedź na wybór preferencji</TITLE>
</HEAD>
<BODY>
Hmmm
<?php
print("$Candy! ");
if ($Candy == "orzeszki ziemne")
  print("Istnieje kilka świetnych rodzajów lodów zawierających małe
         lub połamane $Candy.");
else
{
  print("Myślę, że jeszcze nie istnieją lody zawierające $Candy, ");
  if($Candy == "snickers")
    print("ale czy próbowałeś loda $Candy ?");
  elseif($Candy == "krówki")
    print("ale bardzo potrzebne są lody zawierające $Candy.");
}
?>
</BODY>
</HTML>
```

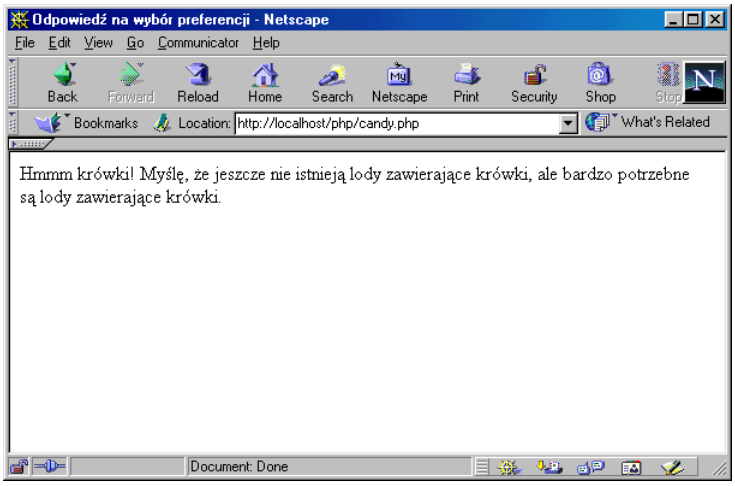
Skrypt ASP realizujący te same funkcje (aby go użyć zmień argument `ACTION` formularza na `candy.asp`):

```
<HTML>
<HEAD>
<TITLE>Odpowiedź na wybór preferencji</TITLE>
</HEAD>
<BODY>
Hmmm
<% Candy = Request.Form("Candy") %>
```

```
<%= Response.Write (Candy)%>
<% If Candy = "orzeszki ziemne" Then %>
Istnieje kilka świetnych rodzajów lodów zawierających małe lub połamane
<% Response.Write(Candy) %>.
<% Else %>
Myślę, że jeszcze nie istnieją lody zawierające <%
Response.Write(Candy) %>,
<% End If %>
<% If Candy = "snickers" Then %>
ale czy próbowałeś loda <% Response.Write(Candy) %> '?
<% Else If Candy = "krówki" Then %>
ale bardzo potrzebne są lody zawierające <%
Response.Write(Candy) %>.
<% End If %>
<% End If %>
</BODY>
</HTML>
```

Wynik jest za każdym razem taki sam jak na rysunku 12.3.

**Rysunek 12.3.** *Wynik użycia PHP lub ASP*




### Używanie GET i POST

Czy wiesz, że PHP może używać zarówno `GET`, jak i `POST` jednocześnie na jednej stronie? Możesz rzucić się w wir dynamicznego tworzenia formularzy!

Szybko pojawia się pytanie: co się stanie, jeżeli (rozmyślnie lub nie) użyję tej samej nazwy zmiennej zarówno w tablicach `GET`, jak i `POST`. PHP przechowuje zmienne `GET`, `POST` oraz `COOKIE` w tablicach o nazwach `$HTTP_GET_VARS`, `$HTTP_POST_VARS` oraz `$HTTP_COOKIE_VARS`, jak również w tablicy `$GLOBALS`. Jeśli wystąpi konflikt, rozwiązywany jest przez nadpisywanie wartości zmiennych w kolejności ustalonej przez opcję `gpc_order` z pliku *php.ini* (należy ustawić również opcję `track_vars`). Późniejszy wygrywa nad wcześniejszym, więc jeżeli użyjesz ustawienia domyślnego `GPC`, *cookie* wygrają z `POST`, a te z `GET`. Możesz zarządzać kolejnością nadpisywania zmiennych, zmieniając kolejność tych trzech liter w odpowiednim wierszu pliku *php.ini*.

Ponieważ PHP posiada zdolność automatycznego przypisywania zmiennych, skrypt PHP pisze się szybciej. W skrypcie PHP po prostu używamy zmiennej bez konieczności pobierania jej wartości lub ponownej definicji. Przykład, jaki podaliśmy, jest trywialny, ponieważ przekazywana jest tylko jedna zmienna. Jednak przy bardziej złożonych formularzach z wieloma zmiennymi lub w przypadku serii formularzy współdzielących wiele zmiennych PHP pozwala zaoszczędzić wiele czasu.

W skrypcie ASP należy odczytać każdą zmienną przysłaną za pomocą metody `POST`, w PHP nie jest to konieczne. W przypadku ASP należy użyć `Request.Form` w przypadku metody `POST` oraz `Request.QueryString` w przypadku metody `GET`. Jeżeli zmienisz metodę przesyłania danych, musisz wszędzie zmienić nazwę kolekcji, co może być źródłem błędów.

Automatyczne przypisywanie zmiennych w PHP również może być źródłem konfliktów nazw zmiennych. PHP po prostu zamieni starą wartość zmiennej na nową. Możesz sterować kolejnością zapisywania zmiennych za pomocą opcji `GPC` z pliku *php.ini*. Najlepszą metodą jest jednak unikanie konfliktów i nadawanie dobrych, zróżnicowanych nazw zmiennych. Większość użytkowników PHP uważa, że użyteczność funkcji automatycznego przypisywania zmiennych przewyższa zagrożenie potencjalnymi konfliktami nazw zmiennych.

# Podsumowanie

Protokół HTTP jest protokołem bezstanowym. Oznacza to, że sam HTML nie potrafi wymieniać danych pomiędzy stronami witryny WWW. Może zostać użyty do przekazywania wartości, ale zewnętrzny program obsługi formularza musi wysilić się, aby przekazać wartość. PHP jest prawdopodobnie najbardziej naturalnym programem obsługi formularza.

Dane można przekazywać za pomocą trzech podstawowych metod: `GET`, `POST` oraz `cookie` (którymi zajmiemy się w rozdziale 26.). Metoda `GET` jest używana do tworzenia złożonych adresów URL, stosowanych wraz z szablonami w dynamicznych witrynach. Nie jest to jednak metoda zalecana do obsługi formularzy, w przeciwieństwie do `POST`.