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

Flash 5.
Kompendium programisty



Autor: P.S. Woods
Tłumaczenie: Marek Binkowski, Rafał Jońca, Robert Pasternak
ISBN: 83-7197-588-0
Tytuł oryginału: Flash 5. Developer's Guide
Format: B5, stron: 258
Zawiera CD-ROM

Ta książka to wspaniałe źródło wiedzy dla twórców flashowych chcących zapoznać się z zagadnieniami programistycznymi (i nie tylko) związanymi bezpośrednio z Flashem. Nie znajdziesz w niej jednak opisu języka ActionScript – celem jest przedstawienie programów i języków programowania, z których zapewne będziesz chciał skorzystać w trakcie kreowania nowej witryny flashowej.

Ponieważ wierzę, że najlepszą metodą nauki jest przeczytanie krótkiego wprowadzenia do tematu, później dokładne omówienie prostego zagadnienia i powolne przechodzenie do coraz to bardziej złożonych pojęć, taką właśnie technikę będę stosował podczas omawiania każdego tematu w każdym rozdziale.

Każdy rozdział dotyczy konkretnego tematu. Treść każdego z nich zaczyna się od przedstawienia prostych, ogólnych przykładów i powoli przechodzi do bardziej złożonych problemów. Można powiedzieć, że początek rozdziału ma Ci dać solidne podstawy danego tematu, abyś dokładnie zrozumiał późniejsze przykłady.

Praktycznie wszystkim pojęciom w książce towarzyszą przykłady. Niektóre z nich są zabawne i bardzo proste, ale głównym ich zadaniem jest ilustracja omawianego zagadnienia. Kod każdego przykładu wraz z omówieniem znajdziesz nie tylko na CD-ROM-ie, ale i w książce. W ten sposób możesz mieć go przed oczami, a na ekranie komputera oglądać jego działanie.

„Flash 5. Kompendium programisty” kierowana jest przede wszystkim do bardziej zaawansowanych twórców flashowych chcących wykorzystać na swych witrynach inne języki skryptowe niż ActionScript. Książka nie jest przeznaczona dla początkujących, ponieważ nie zawiera opisu podstawowych funkcji i narzędzi Flasha. Tę tematykę pozostawiam innym publikacjom. Jeśli czujesz się pewnie w takich zagadnieniach jak: animacja, rysowanie, importowanie obrazów i chcesz się dowiedzieć więcej na temat zaawansowanego wykorzystania filmów Flasha, to jest to książka dla Ciebie.

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

Spis treści

O Autorze 11

Wstęp 13
Do kogo kierowana jest ta książka? 14
Co znajdziesz w książce? 14

Rozdział 1. Wprowadzenie do języka JavaScript 15
Trochę historii 16
Definicja JavaScript 17
Skrypt uruchamiany na komputerze użytkownika 17
Czytelność kodu 18
Prosty i elastyczny język 18
Znaczniki HTML i okienka z komunikatem ostrzegawczym 19
Zmienne 21
Typy danych 22
Sterowanie wykonywaniem skryptu 25
Wyrażenie warunkowe if, else if, else 25
Pętla while 28
Pętla do while 28
Pętla for 29
Struktura decyzyjna switch 30
Operatory 30
Tablice 33
Tablica numerowana 33
Tablica przyporządkowująca 34
Wielowymiarowe tablice 34
Funkcje 35
Bardziej złożone wykorzystanie funkcji 36
Funkcję wbudowane w język JavaScript 37
Narzędzia dotyczące języka JavaScript 39
Programy Note Tab Pro i Edit Plus 39
Program 1st Page 2000 40
Program Dreamweaver 41
Co powinieneś zapamiętać z tego rozdziału 42

Rozdział 2. Język JavaScript — część II 43
Wprowadzenie do programowania obiektowego 43
Obiekt 44
Właściwość i metoda 45
Konstruktory 46
Instrukcja with 47
Funkcje obiektu String 48
Wykorzystywanie obiektów wbudowanych w język JavaScript 52
Obiekt Object 52
Obiekt Math 53
Obiekt Date 55

Obiektowy model dokumentu w języku JavaScript ... 56
Struktura ... 57
Notacja kropkowa ... 57
Manipulacja oknami ... 58
Otwieranie okna ... 58
Opcje okna ... 59
Nowe okno może być obiektem ... 61
Cookies ... 62
Obiekt Cookie — zapisywanie i odczytywanie informacji ... 62
Dalsze rozważania na temat cookie ... 66
Co powinieneś zapamiętać z tego rozdziału ... 66

Rozdział 3. Flash, HTML i przeglądarka ... 67
Pomosty pomiędzy aplikacjami ... 68
Moduły rozszerzające a kontrolki ActiveX ... 68
Wymagania osadzonego obiektu multimedialnego ... 70
Łączenie Flasha z przeglądarką ... 70
Akcje FSCommand i metody filmu Flasha ... 72
Komunikacja pomiędzy oknami ... 73
Pamięć filmu Flasha po stronie klienta w postaci cookies ... 77
Kompatybilność ... 79
Flash i HTML ... 80
Atrybuty znaczników <OBJECT> i <EMBED> ... 80
Szablony ... 82
Integracja z programem Dreamweaver ... 86
Wykrywanie przeglądarki i pluginu ... 86
Podstawowa idea ... 87
Rozwiązanie komercyjne ... 87
Co powinieneś zapamiętać z tego rozdziału ... 90

Rozdział 4. Flash i serwer WWW ... 91
Zasada działania serwera ... 92
Model klient-serwer ... 92
Protokół HTTP ... 94
Typy MIME ... 95
Rodzaje serwerów ... 96
Jakie zadania najlepiej powierzyć serwerowi? ... 97
Dlaczego serwer Apache? ... 98
Instalacja serwera Apache w systemie Windows ... 98
Pobranie Apache'a dla systemu Windows ... 99
Czy Apache jest bezpłatny? ... 101
Pakiet instalacyjny PHPTriad ... 101
Konfiguracja i uruchamianie serwera Apache ... 101
Plik konfiguracyjny httpd.conf ... 102
Plik mime.types ... 104
Polecenia z linii komend ... 104
Ograniczanie dostępu i bezpieczeństwo na serwerze Apache ... 105
Konfiguracja ... 105
Plik indeksowy w każdym katalogu ... 106
Plik .htaccess ... 106
Co powinieneś zapamiętać z tego rozdziału ... 107

Rozdział 5. Język PHP ... 109
Historia języka PHP ... 109
Instalacja i konfiguracja ... 111
Podstawy języka PHP ... 112
PHP i HTML ... 114
Składnia ... 114
Wyświetlanie danych ... 115
Struktury sterujące ... 117
Kilka przydatnych elementów wbudowanych w język PHP ... 118
Zmienna $HTTP_USER_AGENT ... 118
Zmienne $HTTP_POST_VARS i $HTTP_GET_VARS ... 119
Zmienna $PHP_SELF i odmiana polecenia echo ... 120
Funkcja include i przypisywanie zmiennych Flasha bez użycia języka ActionScript ... 122
Wyszukiwanie wzorca w ciągu znaków ... 124
Wyrażenia regularne ... 124
Sprawdzanie poprawności podanego adresu e-mail ... 128
Obsługa plików ... 130
Łączymy wszystko razem — zapisywanie się na listę e-mailową ... 132
Co powinieneś zapamiętać z tego rozdziału ... 133

Rozdział 6. MySQL ... 135
Historia i wprowadzenie ... 135
Jak MySQL wypada w statystykach? ... 137
Model klient-serwer w MySQL ... 138
Instalacja ... 139
Pobranie i instalacja MySQL ... 139
Program Mysqlshow ... 139
Licencja ... 140
Język SQL ... 141
Zaczynamy ... 141
Logowanie do MySQL ... 142
Przydatne polecenia ... 143
Odczytywanie rekordów za pomocą skryptów PHP ... 147
Wyprowadzanie danych z bazy na stronę HTML ... 147
Wyświetlanie danych z bazy w filmie Flasha ... 149
Modyfikacja rekordów — aplikacja administratora ... 151
Łączymy wszystko razem — pocztówki ... 153
Zacznijmy od końca — pocztówka odbiorcy ... 154
Wysyłanie pocztówki ... 156
Co powinieneś zapamiętać z tego rozdziału ... 162

Rozdział 7. Język XML ... 163
Ogólne wiadomości o XML ... 163
Szum informacyjny a obietnice ... 163
Podstawy ... 165
XML we Flashu ... 170
Ogólne informacje ... 170
Statyczne dokumenty XML ... 171
Łącza XMLSocket ... 183
Co powinieneś zapamiętać z tego rozdziału ... 186

Rozdział 8. Program Swift-Generator 187
Dlaczego dynamicznie? 187
Flash 3 i dynamiczne dane 188
Zastępowanie obrazków i dźwięków 188
Podstawy użytkowania Swift-Generatora 188
Co to jest Swift-Generator? 189
Język Swift-Script 190
Jeszcze raz tablica ogłoszeń 196
Szafa grająca 199
Skrypty a dynamiczne generowanie plików SWF 201
Inne narzędzia dynamicznie generujące materiały SWF 202
Co powinieneś zapamiętać z tego rozdziału 203

Rozdział 9. Grafika trójwymiarowa 205
Przegląd zagadnień związanych z grafiką 3D 205
Modelowanie 206
Animacja 207
Rendering 208
Narzędzia 208
Wprowadzenie do grafiki trójwymiarowej 3D przy użyciu programu Strata 210
Obiekty parametryczne 211
Widoki 212
Selekcja 214
Narzędzia transformacji 215
Projekt — zbuduj statek kosmiczny 217
Modelowanie podstawowych obiektów w programie Strata 220
Wytłaczanie obiektu 2D 220
Tworzenie brył obrotowych przy użyciu obiektów 2D 221
Powierzchnie Béziera 222
Wytłaczanie wzdłuż ścieżki 225
Siatki zbudowane z wieloboków — grawitacja 226
Metaobiekty 227
Podsumowanie programu Strata 3D 229
Wprowadzenie do programu Swift 3D 229
Konwersja 230
Materiały 230
Transformacja, selekcja i wyświetlanie 231
Zasady animacji w Swift 3D 233
Kamery 235
Prosta animacja postaci 236
Animacje 3D w czasie rzeczywistym z zastosowaniem odpowiedniego scenariusza 237
Prawdziwe 3D z zastosowaniem skryptów 238
Amorphium Pro — ciekawa alternatywa 239
Co powinieneś zapamiętać z tego rozdziału 243

Dodatek A Zasoby sieciowe dotyczące Flasha 245
Programy dotyczące grafiki 3D 245
Swift3D 246
Amorphium 246
Strata Software 246
Blender 246
Witryna Webreference 3D — Rob Polevoi 247

Titoonic — Tomas Landgreen ... 247
Narzędzie do konwersji między formatami — Crossroads ... 247
Curious Labs — Poser ... 247
Winged Edge Technologies — Nendo ... 247
Animation: Master Martina Hasha ... 248
Photomodeler Lite ... 248
Skrypty ActionScript ... 248
Flashowe ćwiczenia — witryna Actionscripts.org ... 248
E-mailowa lista dyskusyjna Flashcoders ... 248
E-mailowa lista dyskusyjna Flasher ... 249
Witryna Ultrashock.com ... 249
Programy do edycji dźwięku ... 249
FASOFT-N-Track Studio ... 249
Produkty firmy Sonic Foundry ... 249
Cool Edit firmy Syntrillium ... 249
Flashowe narzędzia niezależnych firm ... 250
ActionScript Viewer ... 250
Adobe LiveMotion 1.0 ... 250
Flash Turbine firmy BluePacific ... 250
Firma Flash Jester ... 250
Flashtool ... 251
SwiffTOOLS ... 251
Swift-Generator i Swift-MP3 firmy Swift-Tools ... 251
Flix i SWfX firmy Wildform ... 251
Edytory tekstu ... 251
EditPlus ... 251
NoteTab ... 252
Ultra Edit ... 252
1st Page 2000 firmy EvrSoft ... 252
Różne ... 252
WebSpeed Optimizer ... 252
Tablety firmy Wacom ... 252

Skorowidz ... 253

5.

Język PHP

W tym rozdziale:

- Historia języka PHP
- Instalacja i konfiguracja interpretera
- Podstawy języka PHP
- Kilka przydatnych elementów wbudowanych w język PHP
- Wyszukiwanie wzorca w ciągu znaków
- Obsługa plików
- Łączymy wszystko razem — zapisywanie się na listę e-mailową

Nie mam zamiaru tego ukrywać — uwielbiam język PHP. Po ogromnych problemach tak z nauką jak i używaniem języka Perl, na przykład przy szukaniu ścieżki do interpretera (inna na każdym komputerze), dołączaniu zewnętrznych bibliotek, radzeniu sobie z wybredną składnią oraz popularnym na systemach uniksowych edytorem tekstu Vi pracującym w trybie tekstowym (w tamtym czasie nie istniała wersja języka Perl dla systemu Windows), pojawił się wspaniały PHP3. Jest on bardzo prosty, można go używać na różnych platformach, a w wersji 4. jest niewiarygodnie szybki przy niektórych instalacjach.

Historia języka PHP

Najpierw trochę historii. Język PHP został wymyślony w 1994 roku przez Rasmusa Lerdorfa jako domowy projekt pod nazwą Personal Home Page Tools. Lerdorf dodał jeszcze narzędzia do interpretacji formularzy; jest to najczęstszy powód wykorzystywania skryptów w Internecie.

Od tego czasu PHP zaczął być adaptowany przez innych projektantów witryn, którzy szukali efektywnego i łatwego sposobu obsługi formularzy. Jednak język PHP stał się bardzo popularny dopiero wtedy, gdy Zeev Suraski i Andi Gutmans napisali od nowa jego rdzeń (nową wersję języka PHP oznaczono numerem 3.0). Później jeszcze raz od podstaw opracowano rdzeń języka PHP, znacznie go przyspieszając, a nową wersję oznaczono numerem 4. W wersji 4. wprowadzono nowe funkcje, ale programiści nie muszą uczyć się języka od nowa. Poza tym prawie wszystkie skrypty PHP3 działają w języku PHP4 bez potrzeby wprowadzania żadnych poprawek.

Każdy, kto już kiedyś napisał kilka skryptów, bez problemów zrozumie podstawy języka PHP. Jest on idealnym wyborem dla każdego, kto chce wykonywać pewne funkcje dla filmów Flasha po stronie serwera, ponieważ język ten jest bardzo prosty, ma duże możliwości, jest powszechnie używany i ma wyjątkowo przystępną cenę (jest darmowy). Czytając poniższą listę, zrozumiesz, dlaczego język PHP jest idealnym uzupełnieniem filmów Flasha.

- ✧ Istnieje duża społeczność internetowa, która wymienia się kodami źródłowymi.
- ✧ Ilość użytkowników języka ciągle wzrasta.
- ✧ Język jest wkompilowany w około 40 % serwerów Apache, według danych podanych przez witrynę Netcraft (*http://netcraft.com*).
- ✧ Jest powszechnie obsługiwany.
- ✧ Działa na systemach Windows i UNIX bez potrzeby zmiany kodu.
- ✧ Jest bardzo łatwy w nauce.
- ✧ 99% zasad składni języka PHP jest znane z innych języków.
- ✧ Aktualna wersja (4.) została napisana od nowa i wprowadzono w niej kilka rewolucyjnych rozwiązań znacznie poprawiających wydajność.
- ✧ Język obsługuje dużą liczbę baz danych, a przełączenie się między bazami danych jest bardzo proste. Przeważnie interfejs ODBC nie jest wymagany, ale jest obsługiwany.
- ✧ Ma dużą liczbę funkcji operujących na ciągach znaków, które mogą być uzupełnieniem funkcji z Flasha 5.

Czy kiedykolwiek zastanawiałeś się, jak do witryny flashowej dodać zapamiętywanie najlepszych wyników gry, licznik odwiedzin witryny, prostą księgę gości, która nie wymaga bazy danych, formularz wysyłający pocztę niewymagający użycia języka HTML, plik *.accesslog* zawierający informacje o odwiedzających witrynę lub detekcję systemu klienta po stronie serwera? Rozwiązaniem wszystkich tych problemów może być język PHP. Istnieje duże prawdopodobieństwo, że jeśli wymyślisz sobie jakiś cel, będziesz mógł w jedno popołudnie na tyle nauczyć się języka PHP, aby utworzyć proste rozwiązanie problemu (lub przynajmniej znaleźć i zainstalować skrypt z najbardziej odpowiednim rozwiązaniem). W tym rozdziale omówię podstawy języka PHP. Najbardziej mogą się one przydać osobom, które zamierzają łatwo i szybko tworzyć aplikacje serwerowe dla filmów Flasha.

Instalacja i konfiguracja

Jeśli w poprzednim rozdziale zainstalowałeś pakiet PHPTriad, możesz przejść do następnego podrozdziału, ponieważ nie musisz już niczego instalować. Informacja dla użytkowników systemu MacOS: w czasie pisania tego tekstu obiecywano, że system MacOS X będzie obsługiwał podstawowe uniksowe standardy internetowe, czyli PHP. Jeśli jednak już dzisiaj chcesz zainstalować obsługę PHP na komputerze Macintosh, to powinieneś zajrzeć na witrynę WebTen (*www.tenon.com/products/webten*)[1].

Dla użytkowników systemu Windows instalacja obsługi języka PHP będzie bardzo podobna do instalacji aplikacji Apache. Zamierzamy zainstalować wersję CGI języka PHP. Jest także dostępna wersja ISAPI i różne rozszerzenia programistyczne do głównej części języka PHP napisane dla systemu Windows (przeważnie z dołączonym kodem źródłowym). Jednak w tym podrozdziale nie zajmiemy się tym tematem, ponieważ szukamy najprostszej i bezproblemowej instalacji obsługi języka PHP w systemie Windows. Naszym zadaniem jest utworzenie środowiska, w którym będziemy mogli bezproblemowo testować skrypty PHP. Niezbyt nas interesuje wydajność wybranego rozwiązania.

1. Przejdź do części *Downloads* witryny *http://php.net*. Pojawi się strona z odnośnikami do kilku stabilnych wersji aplikacji PHP dla systemu Windows. Znajdź najnowszą wersję dostępną jako instalator i pobierz ją.

2. Uruchom pobrany plik instalacyjny. Instalator zapyta, gdzie ma zainstalować aplikację PHP. Tutaj masz pełną dowolność. Za chwilę dodamy do pliku *httpd.conf* aplikacji Apache ścieżkę do interpretera języka PHP, więc możesz wybrać taki katalog, jaki Ci odpowiada. Instalator zapyta także o serwer wychodzącej poczty (serwer SMTP) i Twój adres e-mail. Dane te są potrzebne do wbudowanej w język PHP funkcji poczty. Na końcu instalator zapyta, z jakim serwerem stron WWW będziesz używał interpretera języka PHP. Wybierz opcję *Apache*. Zakończ instalację.

3. Otwórz plik *httpd.conf* znajdujący się w folderze *conf* katalogu z aplikacją Apache. Edytowaliśmy ten plik w poprzednim rozdziale.

4. Pierwszą rzeczą, jaką powinieneś dodać, jest uwzględnianie pliku *index.php* jako pliku początkowego. Znajdź linię o treści `DirectoryIndex index.html` i poniżej w nowej linii napisz `DirectoryIndex index.php`.

5. Następnie dodaj ścieżkę do katalogu z interpreterem języka PHP. Możesz to zrobić, przypisując ścieżkę do parametru `ScriptAlias`. Dodaj linię o treści `ScriptAlias /php/ "C:/php4/"` łącznie z końcowym ukośnikiem i cudzysłowami. Tekst `"C:/php4/"` zastąp ścieżką do katalogu, w którym zainstalowałeś interpreter. W tym katalogu serwer będzie wykonywał skrypty PHP.

[1] Lub zainstalować system Linux skompilowany pod procesory PowerPC — *przyp. tłum.*

6. Musimy dodać typ MIME dla skryptów PHP. Możesz to zrobić, edytując plik konfiguracyjny *mime.types*, znajdujący się w katalogu *conf* lub dodając następującą linię do pliku *httpd.conf*: `AddType application/x-httpd-php .php`.

7. Na końcu musimy określić i podać serwerowi Apache, który program będzie wykonywał skrypty w plikach o rozszerzeniu PHP. Możesz to zrobić, dodając następującą linię `Action application/x-httpd-php "/php/php.exe"`. Musisz tutaj użyć cudzysłowów.

8. Otwórz plik *php.ini*. Znajdziesz go w katalogu z systemem Windows. Instalator automatycznie umieścił plik w tym miejscu. Poza tym instalator poustawiał wszystkie najważniejsze zmienne, więc właściwie *nie musisz* nic zmieniać. Jest jednak pewna denerwująca funkcja (powiadamianie o błędach), którą warto wyłączyć. Jeśli pozostawisz wartość domyślną, z wbudowanego w interpreter debuggera będziesz otrzymywał masę uwag. Na przykład będziesz otrzymywał ostrzeżenia o tym, że nie używasz zalecanego stylu i składni. Często przypomina to ścisłe oznaczenia z języka Perl. Aby debugger interpretera pokazywał wyłącznie błędy, powinieneś zmienić linię z parametrem `error_reporting` na następującą:

```
error_reporting = E_ERROR;
```

Pełną listę opcji tego parametru znajdziesz na stronie *www.php.net/manual/en/features.error-handling.php*. Wszystkie powyższe ustawienia dotyczą wyłącznie takich instalacji jak nasza (w systemie Windows). Jeśli używasz skryptów PHP na komercyjnym serwerze, nie musisz niczego ustawiać, ponieważ wszystkie parametry (także plik *httpd.conf* serwera Apache) są już ustawione. Jeśli potrzebujesz dedykowanego, a nie wirtualnego serwera, powinieneś zatrudnić specjalistę od aplikacji Apache albo spędzić kilka miesięcy na czytaniu i eksperymentowaniu z optymalną konfiguracją serwera. Chociaż pliki, którymi się zajmowaliśmy, są niewielkie, mają ogromny wpływ na wydajność serwera.
Teraz serwer Apache powinien obsługiwać skrypty PHP.

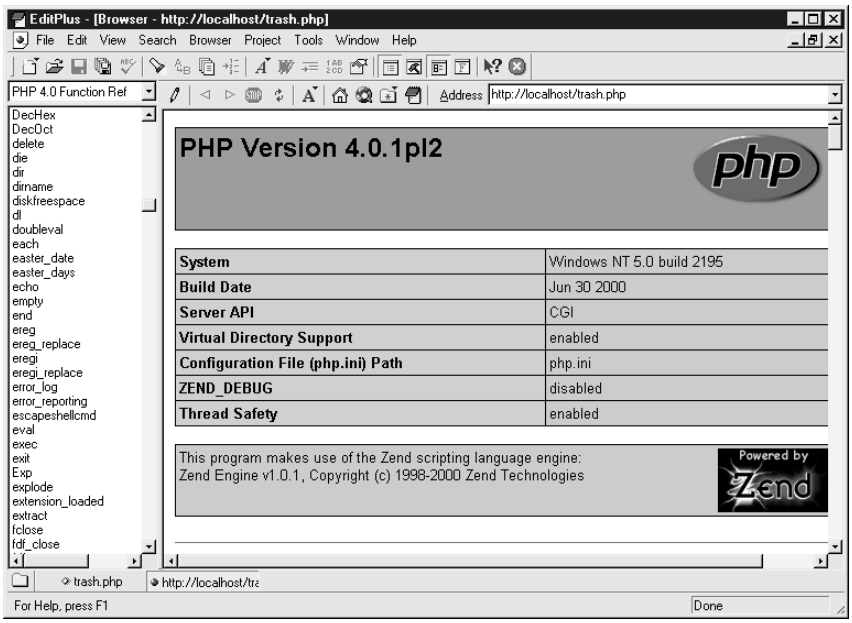
Podstawy języka PHP

Najlepiej sprawdzić, czy wszystko działa prawidłowo, zanim zaczniemy zajmować się innymi sprawami. W edytorze tekstowym utwórz plik HTML zawierający następujący kod:

```
<html>
<body>
<?php
phpinfo();
?>
</body>
</htmlbody>
```

Zapisz plik pod dowolną nazwą, ale z rozszerzeniem PHP (na przykład możesz plik nazwać *information.php*) i otwórz go w przeglądarce. Muszę tutaj wspomnieć o kilku sprawach, które dla jednych są oczywiste, ale inni mogą nie zdawać sobie z nich sprawy.

- ✧ Serwer Apache musi być uruchomiony.
- ✧ Jeśli serwer Apache działał, gdy modyfikowałeś plik *httpd.conf*, będziesz musiał go zrestartować, aby zmiany w pliku zostały uwzględnione. Innymi słowy, musisz wyłączyć i ponownie uruchomić serwer Apache (nie cały komputer).
- ✧ Musisz otworzyć plik za pomocą wywołania HTTP kierowanego do serwera. (Nie możesz po prostu otworzyć w przeglądarce pliku znajdującego się na dysku twardym). Najprościej zrobić to, wpisując w pasku adresowym przeglądarki adres URL — na przykład adres *http://localhost/information.php*[2].
- ✧ Wiele edytorów HTML i tekstowych pozwala na taki podgląd dokumentów, jakby plik znajdował się na serwerze — niektóre z nich pozwalają na to nawet w dodatkowym oknie środowiska edycyjnego. Rysunek 5.1 przedstawia jeden z takich edytorów, program EditPlus (*www.editplus.com*), w którym oglądamy nasz testowy plik w — wbudowanej w program — przeglądarce. Tego rodzaju funkcja jest po prostu nieoceniona przy pisaniu skryptów. Nie tylko zaoszczędza Ci problemów z pamięcią cache przeglądarki (zawsze masz pewność, że oglądasz najnowszą wersję pliku), ale także pozwala się skupić, ponieważ nie musisz się przełączać między różnymi aplikacjami.



Rysunek 5.1. *Edytor EditPlus z podglądem naszego pliku testowego*

[2] Pamiętaj o tym, że w podanym przykładzie plik information.php musi się znajdować w folderze htdocs. Znajdziesz go w katalogu z aplikacją Apache — *przyp. tłum.*

Jeśli wykonałeś wszystkie powyższe kroki, a interpreter PHP nadal nie działa, najlepiej będzie, jeżeli poszukasz w Internecie informacji na temat błędów, jakie otrzymujesz. Witryny *www.zend.com* i *www.phpbuilder.com* zawierają fora dyskusyjne, na których znajdziesz odpowiedzi na pytania zadawane przez początkujących. Jeśli nie chcesz się zajmować samodzielnym rozwiązywaniem pojawiających się problemów, możesz pobrać i zainstalować aplikację PHPTriad, w której wszystko jest już ustawione.

PHP i HTML

Skrypt PHP można umieścić w kodzie HTML (jak w pierwszym przykładzie), ale może to być też skrypt umieszczony w osobnym pliku. Gdy zamierzasz wykorzystywać język PHP jako pomost pomiędzy filmem Flasha a źródłem danych, w zasadzie nie potrzebujesz znaczników języka HTML. Możesz pominąć kod HTML, ponieważ otrzymane dane nie będą wyświetlane przez żadną przeglądarkę. Na przykład, jeśli wynik otrzymywany w pierwszym przykładzie chcielibyśmy przekazać do Flasha, wystarczyłby następujący kod:

```
<?php
phpinfo();
?>
```

Jak widzisz, skrypty PHP są zawsze umieszczane między znacznikami `<?php` i `?>`. Funkcja `phpinfo()` jest specjalną funkcją, która zwraca przydatne informacje o środowisku serwera. Jeśli przyjrzysz się tym, które pokazały się w przeglądarce po otwarciu pierwszego przykładu, znajdziesz wartości zmiennych z plików `php.ini` i `httpd.conf` oraz informacje o samym komputerze.

Składnia

Język PHP zwraca uwagę na wielkość liter, więc przyjmij jeden schemat nazewnictwa i trzymaj się go cały czas. Na przykład, jeśli utworzysz zmienną o nazwie `$monkeySee`, a następnie spróbujesz się do niej odwołać, pisząc `$MonkeySee`, otrzymasz komunikat o błędzie.

Zapewne zauważysz, że składnia języka PHP jest znajoma i łatwa, jeśli wcześniej pisałeś skrypty w języku ActionScript bazującym na standardzie ECMA-286. Funkcje, instrukcje i większość struktur decyzyjnych działa dokładnie tak samo jak w języku ActionScript.

Istnieje jednak jedna duża różnica w sposobie wywoływania metod i właściwości. Nie ma struktury obiektów opartej na notacji kropkowej. Obiekty przyjmują formę prefiksu, po którym znajduje się znak podkreślenia a dopiero po nim nazwa metody. Nazwę klonu obiektu podaje się jako argument metody wbudowanej lub utworzonej za pomocą konstruktora. Porównaj kod znany z języka JavaScript lub ActionScript z kodem, który musiałbyś napisać w języku PHP.

```
//w języku ActionScript
var lista=new Array ("mleko", "jajka", "mąka");
lista.pop();
```

```
//w języku PHP
array_pop($lista);
```

Zmienne

Wszystkie zmienne w języku PHP identyfikuje się, poprzedzając je znakiem dolara ($). Język PHP posługuje się elegancką sztuczkę podobną do funkcji `eval()` języka ActionScript Flasha, która pozwala dynamicznie wskazywać identyfikator zmiennej.

```
$zwierze="pies";
$$zwierze="terier";
```

W drugiej linii tego przykładu używamy *zmiennej dynamicznej* `$$zwierze`, która w tym przypadku jest równoznaczna ze zmienną `$pies`. Bardzo łatwo można wyjaśnić zasadę działania zmiennych dynamicznych: interpreter PHP oblicza wszystkie znaki dolara, zaczynając od prawej strony i idąc w lewo. W tym przykładzie po obliczeniu `$zwierze` otrzymuje wartość `pies`, więc zmienna dynamiczna `$$zwierze` jest w tym przypadku równa zmiennej `$pies`. Możesz tworzyć bardziej złożone struktury zmiennych dynamicznych, ale wtedy użyteczność tej sztuczki maleje, ponieważ coraz trudniej zapanować nad tym, do jakiej zmiennej przypisujemy wartość.

Typy

Język PHP, podobnie jak język ActionScript Flasha, nie jest językiem, w którym musisz jednoznacznie określić, czy dana zmienna przechowuje liczbę, ciąg znaków, tablicę czy obiekt. Możesz dynamicznie zmieniać typ danych przechowywany w zmiennej, na przykład, skonwertować liczbę na ciąg znaków (jak w języku ActionScript), ale do łączenia wyrazów musisz używać znaku kropki zamiast plusa.

```
$zaokraglonaLiczba="Łącznie płacisz " . $zaokraglonaLiczba .
    "\.00 złotych";
```

Wyświetlanie danych

Nie pokazałem jeszcze, jak wyświetlić ciąg znaków lub liczbę w oknie przeglądarki internetowej. Jest to bardzo proste. Istnieją dwa sposoby wyświetlania danych — w postaci sformatowanej lub niesformatowanej.

Polecenie echo

Jeśli chcesz po prostu wyświetlić wartość zmiennej (liczbę lub ciąg znaków), korzystaj z polecenia `echo`. Zapewne zastanawiasz się, jak wyświetlić standardowy (w książkach o programowaniu) tekst „Witaj świecie”. Oto sposób:

```
echo "Witaj świecie<BR>\n";
//wyświetla tekst Witaj świecie
$radosnaWiadomosc=" Witaj świecie<BR>\n";
echo $radosnaWiadomosc;
// to samo
echo "$radosnaWiadomosc";
//nadal wyświetla tekst Witaj świecie
```

W powyższym kodzie powinieneś zauważyć kilka rzeczy. Po pierwsze, komentarze oznacza się tak samo jak w języku ActionScript. Komentarze można także pisać inaczej — znak funta lub „#" ukrywa przed interpreterem wszystko za tym znakiem aż do końca tej linii.

W powyższym przykładzie zauważyłeś, że wartości tekstowe działają tak samo jak w języku ActionScript Flasha. Jeśli masz ciąg znaków zawarty między cudzysłowami, jest on traktowany jak wartość tekstowa, natomiast identyfikator zmiennej poza cudzysłowami jest interpretowany. Istnieje jednak pewna różnica: zmienna jest interpretowana nawet wtedy, gdy znajduje się wewnątrz cudzysłowów.

Oczywiście znacznik `<BR>` powoduje przejście do następnej linii w kodzie HTML, ale powód umieszczenia znaku nowej linii nie jest już tak oczywisty. Jeśli korzystasz z języka PHP na stronie HTML, całkiem możliwe, że będziesz chciał zobaczyć wygenerowany przez skrypt kod HTML (na przykład, gdy pojawią się problemy z formatowaniem). Znacznie łatwiej zrozumieć kod źródłowy, gdy każdy znacznik znajduje się w osobnej linii.

Funkcja printf

Przypuśćmy, że zamierzasz utworzyć ładnie sformatowaną tabelę HTML zawierającą liczby od 1 do 12, ich podwojenie i kwadrat. Podwojone niewielkie liczby i ich kwadraty będą kolumnami w bazie danych tworzonej w następnym rozdziale, wiec nie zlekceważ tego ćwiczenia.

Mógłbyś w tym przykładzie skorzystać z polecenia `echo`, ale na środku pętli otrzymałbyś wielką niełądną linię kodu. Chociaż osoby piszące w języku ActionScript są przyzwyczajone do długich linii kodu, nie było to normą w starszych i elegantszych językach skryptowych. Oto bardziej wyrafinowana wersja napisana z wykorzystaniem funkcji `printf`:

```
echo "<table>\n";
echo "<tr><td>Liczba</td><td>Podwojenie</td><td>Kwadrat</td></tr>\n";
for($i=1;$i<=12;$i++){
  printf("<tr><td>%s</td><td>%s</td><td>%s</td></tr>\n",
    $i, $i*2, $i*$i);
}
echo "</table>\n";
```

Funkcja `printf` używa w pierwszym argumencie dowolnej liczby znaczników `%s` oraz ciągu znaków w dowolnym formacie. Liczba argumentów podanych po pierwszym argumencie musi odpowiadać liczbie znaczników `%s` umieszczonych w pierwszym argumencie, ponieważ funkcja (idąc po kolei od lewej strony) zastępuje znaczniki `%s` wartościami podanymi jako kolejne argumenty. Innymi słowy, gdy funkcja znajdzie pierwszy znacznik `%s`, zastępuje go drugim argumentem — w tym przykładzie będzie to `$i`. Gdy znajdzie następny znacznik `%s`, zastępuje go kolejnym argumentem — w tym przypadku `$i*2`.

Jak zapewne zauważyłeś, funkcja ta jest bardzo przydatna, gdy tworzysz kod HTML w skrypcie. Jeśli odseparujemy dynamiczne dane od formatowania, możemy bardzo łatwo zmieniać formatowanie.

Struktury sterujące

Większość struktur sterujących języka PHP wygląda dokładnie tak samo jak w językach JavaScript lub ActionScript, jednak istnieją pewne różnice oraz kilka nowych elementów. Pierwsza z różnic dotyczy wyrażenia warunkowego `if...else if`.

```
if($cos="kewl"){
  idzDoSzkoly();
} elseif($cos="nachos"){
  idzDoSklepu();
}
```

Widzisz różnicę? Nie ma spacji między `else` i `if`. Tak, ale jest jeszcze jedna różnica, której zapewne nie zauważyłeś. W języku PHP operator porównania to jeden, a nie dwa znaki równości.

Inną przydatną strukturą sterującą języka PHP jest pętla, którą w języku ActionScript wywoływałeś instrukcją `for...in`. W języku PHP, jak i wcześniej w języku Perl, ten rodzaj pętli wykonuje się za pomocą instrukcji `foreach`. Poniżej przedstawiam porównanie zapisu tego rodzaju pętli w językach ActionScript i PHP.

```
// w języku ActionScript Flasha
for(skladnik in lista){
  trace(lista[skladnik] + "\n");
}

// w języku PHP
foreach ($lista as $skladnik){
  echo "$skladnik<BR>\n";
}
```

W tym przykładzie wypisywaliśmy *wartość* każdego elementu tablicy `lista` za pomocą dodatkowej zmiennej. Istnieje także sposób pobierania identyfikatora każdego elementu tablicy i umieszczania go w tymczasowej zmiennej, gdy pętla przechodzi przez tablicę. Dla tradycyjnej tablicy byłaby to po prostu seria liczb, ale dla tablicy przyporządkowującej będzie to nazwa pary nazwa-wartość. Jest to bardzo ważne, jeśli zamierzasz wydobywać pary nazwa-wartość z formularza.

```
foreach($lista as $skladnik => $odmiana){
  printf("%s:  %s<BR>\n".$skladnik,$odmiana);
}
```

W tym przykładzie `$skladnik` to tymczasowa zmienna przechowująca identyfikator (nazwę) kolejnych elementów tablicy, a zmienna `$odmiana` przechowuje wartość kolejnych elementów tablicy. Często programiści nazywają te zmienne `$klucz => $wartosc` (`$key => $value`), aby nie pomylić się, która z nich zawiera identyfikator, a która wartość. Ja jednak wolę nadawać odpowiednie opisowe nazwy w zależności od sytuacji.

Struktura decyzyjna `switch` działa dokładnie tak samo jak w języku JavaScript. W zasadzie w poniższym przykładzie z rozdziału 6. nie musiałem modyfikować struktury decyzyjnej `switch`; dodałem tylko znak dolara przed nazwami zmiennych i zastąpiłem polecenie `document.write()` poleceniem `echo`.

```
switch ($haslo){
case 'TNPRock':
  echo "Kyles_Home_Page";
  break;
case 'SPRocks':
  echo "Stans_Home_Page";
  break;
default:
  echo "Zapomniałeś hasła?";
  break;
}
```

Kilka przydatnych elementów wbudowanych w język PHP

Jedną z rzeczy, jakie lubię w języku PHP, jest to, że często przyziemne zadania (zazwyczaj wykorzystywane) zostały zawarte w specjalnych funkcjach lub można je napisać znacznie krócej. Wygląda na to, że osoby projektujące ten język dokładnie wiedziały, jakie zadania najczęściej wykonują programiści i maksymalnie je uprościli, zwykle do jednego słowa lub funkcji. W tej części rozdziału przyjrzymy się kilku bardzo użytecznym funkcjom.

Zmienna $HTTP_USER_AGENT

W rozdziale 2. dotyczącym języka JavaScript omówiliśmy pokrótce obiekt `navigator`. Obiekt ten zawierał szczegółowe informacje na temat przeglądarki użytkownika, przede wszystkim jej rodzaj i wersję. Informacje te są często wykorzystywane w skryptach interpretowanych na komputerze użytkownika, czasem nawet do detekcji zainstalowanych modułów rozszerzających. Chociaż jest to bardzo pomocy obiekt w obiektowym modelu dokumentu, każda znana implementacja języka JavaScript używana w przeglądarkach jest beznadziejna w kategoriach zachowania standardu. Możesz albo pisać skrypty działające na z góry określonych przeglądarkach, albo napisać strasznie długi skrypt próbujący działać na wszystkich przeglądarkach (próbujący, ponieważ są nikłe szanse na to, że uda się taki skrypt napisać).

Rozwiązaniem dla prostych zadań (na przykład detekcja wersji przeglądarki) może być wbudowana w język PHP zmienna `$HTTP_USER_AGENT`. Jest to zmienna *globalna*, więc już wiesz, gdzie możesz ją znaleźć. Jest zawsze dostępna i jest zdefiniowana, jeśli skrypt został wywołany przez przeglądarkę.



Ostrzeżenie

Przykłady dla tego rozdziału musisz umieścić w ogólnodostępnym katalogu na serwerze Apache. Zalecam skopiowanie całego katalogu z przykładami (na CD-ROM-ie jest to katalog *Rozdzial_5*) do katalogu *htdocs*. Gdy będziesz chciał zobaczyć, jak działa dowolny przykład, wpisz w polu adresowym przeglądarki *http://localhost/Rozdzial_5/nazwa_przykladowego.pliku*.

Zmienne $HTTP_POST_VARS i $HTTP_GET_VARS



Pliki *echoVarsFromURLString.php*, *echoVarsFromHTMLForm.php*, *genericForm.html*, *submitInRecursiveForm.fla*, *submit.html* i *submit.fla* z katalogu *Rozdzial_5* na CD-ROM-ie.

Obsługa formularzy była jednym z najważniejszych zadań języka PHP i w każdej kolejnej wersji jest coraz lepsza. Poniższy kod (znajdziesz go pliku *echoVarsFromURLString.php* na CD-ROM-ie) przetwarza zmienne podane po adresie URL i wypisuje je w formacie jedna para w jednej linii. Umieść cały katalog *Rozdzial_5* z CD-ROM-u na serwerze Apache (patrz poprzednia ramka z ostrzeżeniem). W pola adresowym przeglądarki wpisz wywołanie HTTP zawierające kilka zmiennych, na przykład: *http://localhost/Rozdzial_5/echoVarsFromURLString.php?imie=Marek&praca=OK*.

```
foreach($HTTP_GET_VARS as $key => $value){
  printf("%s:   %s<BR>\n",$key,$value);
}
```

Powinieneś znać już wszystkie polecenia tego skryptu poza wbudowaną w język PHP zmienną `$HTTP_GET_VARS`. Jest to w zasadzie tablica o globalnym zasięgu, podobnie jak zmienna `$HTTP_USER_AGENT`.

Używając zmiennej środowiskowej `$HTTP_GET_VARS`, musisz pamiętać, że zawiera ona tylko te zmienne formularza HTML, które zostały wysłane za pomocą metody GET (wraz z wywołaniem HTTP). Innymi słowy, jeśli wywołasz ten skrypt przy użyciu formularza HTML korzystającego z metody POST, nie będzie działał.

Istnieje osobna zmienna środowiskowa, która obsługuje pary zmienna-wartość wysłane za pomocą metody POST. Jeśli umieściłeś wszystkie pliki do tego rozdziału na serwerze Apache, możesz w polu adresowym przeglądarki napisać *http://localhost/Rozdzial_5/genericForm.html*.

```
foreach($HTTP_POST_VARS as $key => $value){
  printf("%s:   %s<BR>\n",$key,$value);
}
```

Gdy wypełnisz formularz i klikniesz przycisk *Wyślij do PHP*, zostanie wywołany plik *echoVarsFromHTMLForm.php* ze skryptem PHP, dokładniej wywoła go znacznik HTML `<form method="post" action="echoVarsFromHTMLForm.php">`. Powinieneś zobaczyć tak samo sformatowaną stronę jak w skrypcie używającym metody GET protokołu HTTP.

Następny przykład: plik *submit.html* wykonuje to samo zadanie, ale za pomocą filmu Flasha. Poniższy kod znajdziesz w pliku *submit.fla*. Wyeksportowany film Flasha *submit.swf* został osadzony w pliku *submit.html*.

W poniższym kodzie nie użyłem funkcji `escape()`, aby był bardziej czytelny. Może to spowodować pewne problemy w niektórych przeglądarkach, gdy użytkownik w pola formularza wpisze znaki specjalne (także polskie znaki diakrytyczne) lub spacje. Kolejny przykład: plik *submitInRecursiveForm.fla* rozwiązuje ten problem. Jeśli chcesz, możesz otworzyć ten plik aby przeanalizować zawarty tam skrypt.

```
on (release, releaseOutside, keyPress "<Enter>") {
  bigURLString = "echoVarsFromURLString.php?submit=ohYes&variable01="+variable01+
  "&variable02="+variable02+"&variable03="+variable03+"&variable04="+
  variable04+"&variable05="+variable05;
  getURL (bigURLString);
}
```

Ponieważ skrypt ten używa akcji `getURL` (wywołanie HTTP wykorzystujące metodę GET, podobnie jak w przeglądarce), używamy skryptu PHP odczytującego zmienne z tablicy `$HTTP_GET_VARS`. Skrypt ten znajduje się w pliku *echoVarsFromURLString.php*.

Z tego typu komunikacji ze skryptami serwerowymi korzysta się rzadko. Częściej używa się akcji `loadVariables`, ponieważ pozwala ona wyświetlić dane otrzymane ze skryptu w filmie Flasha. Jednak wykorzystywanie metody GET w prostych zadaniach (na przykład przesyłanie kilku zmiennych w adresie URL) ma wiele zalet, między innymi łatwiejszą obsługę błędów (na serwerze), a użytkownikowi daje większą kontrolę (pozwala się cofnąć). W następnych przykładach zobaczysz, jak można użyć tej metody do dynamicznego osadzania różnych filmów na stronie HTML.

Zmienna $PHP_SELF i odmiana polecenia echo



Plik *recursiveForm.php* z katalogu *Rozdzial_5* na CD-ROM-ie.

Następnym logicznym krokiem jest połączenie formularza ze skryptem PHP. Umieść plik *recursiveForm.php* w folderze *Rozdzial_5* katalogu *htdocs* i wczytaj go do przeglądarki z serwera. Poniższy kod odpowiada w tym przykładzie za sterowanie działaniem skryptu.

```
if($submit){
  foreach($HTTP_POST_VARS as $key => $value){
    printf("%s:   %s<BR>\n",$key,$value);
  }

//w przeciwnym przypadku, wyświetl formularz
}else{...
```



Uwaga

Jeśli masz włączoną opcję pokazywania niekrytycznych błędów, otrzymasz komunikat o błędzie w pierwszej linii. Jeśli nie chcesz widzieć takiego komunikatu lub używasz komercyjnego serwera, który wyświetla informacje o błędach (zdarza się to rzadko), zainicjalizuj zmienną (`$submit = ""`), a następnie zmień testowany warunek na `if($submit != "")` lub inny działający tak samo.

Skrypt wydobędzie zmienne otrzymane metodą POST tylko wtedy, gdy zmienna `$submit` ma przypisaną dowolną wartość. Przycisk wysyłania danych z formularza powoduje przypisanie wartości do tej zmiennej, ponieważ w znaczniku dodaliśmy atrybut `name`: `<input type="submit" name="submit" value="Wyślij do PHP">`.

Jeśli zmienna `$submit` ma przypisaną wartość, oznacza to, że użytkownik kliknął już wcześniej przycisk *Wyślij do PHP*. Aby dowiedzieć się, jak do tego mogło dojść, musimy przeanalizować drugi fragment skryptu, czyli następujący kod:

```
}else{
  echo<<<ENDPAGE
  <form method="post" action="$PHP_SELF">
  <input type="text" name="variable01" value="variable01">
  <input type="text" name="variable02" value="variable02">
  <input type="text" name="variable03" value="variable03">
  <input type="text" name="variable04" value="variable04">
  <input type="text" name="variable05" value="variable05">
  <input type="submit" name="submit" value="Wyślij do PHP">
  </form>
ENDPAGE;
}
```

Argument `action` w tym formularzu jest równy zmiennej `$PHP_SELF`, co w języku PHP oznacza „wyślij wywołanie HTTP metodą POST z powrotem do tego skryptu”. W tym przykładzie jest to równoznaczne przypisaniu argumentowi `action` wartości `recursiveForm.php` lub `http://localhost/recursiveForm.php`.

Nowością w tym kodzie jest także argument polecenia `echo`. Jeśli po słowie echo występują trzy znaki mniejszy od (`<<<`), ciąg znaków po nich zastępuje cudzysłowy jako ograniczniki działania polecenia `echo`. Tego rodzaju polecenie jest nazywane *tutaj dokument*. Zapewne zauważyłeś, dlaczego tego rodzaju polecenie jest takie ważne — tekst, który chcemy wypisać zawiera wiele cudzysłowów. Używając sposobu *tutaj dokument*, musisz jednak pamiętać o kilku sprawach:

- ✧ definicja ogranicznika rozpoczyna się zaraz za ostatnim znakiem <, łącznie ze spacjami;
- ✧ końcowy ogranicznik musi się znajdować w osobnej linii zakończonej średnikiem (jest to koniec rozpoczętego wcześniej polecenia `echo`);
- ✧ obydwa ograniczniki muszą być *identyczne* (wielkość liter ma znaczenie);
- ✧ ogranicznik nie jest zmienną (brak znaku dolara na początku);

- jeśli jako ogranicznik wybierzesz słowo, które może wystąpić w wypisywanym ciągu znaków, polecenie `echo` może się w takim miejscu zakończyć, powodując nieprzewidziane rezultaty. Zawsze wybieraj słowo, które nie może pojawić się w wypisywanym tekście.

Skoro wiesz już, jak działają poszczególne fragmenty skryptu, przyjrzyj się skryptowi jako całości, aby zobaczyć, jak pracuje. Gdy użytkownik po raz pierwszy wczyta stronę, zmienna `$submit` nie jest zdefiniowana, więc wykona się druga część skryptu — wyświetlanie formularza. Jeśli przyjrzysz się kodowi źródłowemu HTML, używając funkcji podglądu kodu źródłowego w przeglądarce, zobaczysz, że zmienna `$PHP_SELF` została zastąpiona ścieżką do pliku ze skryptem. Gdy użytkownik wypełni formularz i kliknie przycisk *Wyślij do PHP*, dane z formularza zostaną przesłane do tego samego skryptu. Teraz zmienna `$sumbit` jest zdefiniowana (ma wartość), więc wykona się pierwsza część skryptu powodująca wydrukowanie par zmienna-wartość.

Funkcja include i przypisywanie zmiennych Flasha bez użycia języka ActionScript



Pliki *recursiveForm-swf.php*, *SWFform.inc*, *submitInRecursiveForm.swf*, *SWFDisplay.inc* z katalogu *Rozdzial_5* na CD-ROM-ie.

Następny przykład, plik *recursiveForm-SWF.php*, działa tak samo jak poprzedni przykład, ale interfejs stanowi film Flasha. Kod tego skryptu (poniżej) jest zdecydowanie krótszy.



Uwaga

Przykłady w tym rozdziale zostały tak napisane, aby jak najbardziej ułatwić ich zrozumienie. Wiele znaczników HTML zostało usuniętych, aby nie zaśmiecać kodu. Mogłem to zrobić, ponieważ dzisiejsze przeglądarki są bardzo tolerancyjne i pokazują te pliki bez żadnych błędów. Jeśli jednak tworzysz skrypty PHP, które chcesz umieścić na witrynie, zawsze powinieneś stosować znaczniki HTML odpowiednie dla przeglądarki wykorzystywanej przez użytkowników.

```
//jeśli zmiennej $submit przypisano wartość (użytkownik wysłał
//formularz)...
if($submit){
  foreach($HTTP_GET_VARS as $key => $value){
    $nameValPairs.=("$key=$value&");
  }
  include ("SWFDisplay.inc");

//w przeciwnym przypadku, wyświetl formularz
}else{
  include ("SWFform.inc");
}
```



Uwaga

Pliki z rozszerzeniem INC muszą znajdować się w tym samym katalogu, co plik ze skryptem PHP, aby skrypt działał prawidłowo. Jeśli z jakiegoś powodu musisz je umieścić w innym katalogu, pamiętaj, aby umieścić ścieżkę do tego katalogu w argumencie funkcji `include()`.

Skrypt nadal działa tak samo, ale zamiast formularza HTML i pętli `foreach` wyświetlającej pary zmienna-wartość możesz znaleźć funkcję `include()`. Funkcja ta przyjmuje tylko jeden argument — nazwę pliku. Skrypt umieszcza podany plik w wyjściowym kodzie HTML. W tym przykładzie używamy kilku plików, aby nie tworzyć jednego długiego skryptu zawierającego dwa osadzenia plików .SWF. Nazwa rozszerzenia nie ma szczególnego znaczenia — możesz dołączać pliki HTML, TXT lub nawet PHP. Plik *SWFform.inc* nie zawiera niczego szczególnie ciekawego, ponieważ powoduje osadzenie filmu *submitInRecursiveForm.swf*. Film używa metody GET protokołu HTTP, jak w poprzednim przykładzie z plikiem SWF.

Bardziej interesująca jest zawartość pliku *SWFDisplay.inc*. Poniższy fragment kodu ukrywa się w połowie pliku.

```
<PARAM NAME=movie VALUE="echoVars.swf?<?php echo $nameValPairs ?>">
```

Jest to głównie znacznik `<PARAM>` zagnieżdżony wewnątrz znacznika `<OBJECT>`. Jednak po prawej stronie możesz zauważyć króciutki skrypcik PHP. Wykona się on, gdy główny skrypt będzie wyprowadzał ten plik na wyjście. Plik ten jest interpretowany jak plik HTML, więc znajdujący się wewnątrz niego skrypt PHP zostanie wykonany. Chociaż może wydawać się to trochę zagmatwane, podobne sztuczki stosuje się bardzo często (choć raczej nie w przypadku stron z filmami Flasha).

Zmiennej `$nameValPairs` nadaliśmy wartość w pętli `foreach`. Zawiera ona ciąg znaków zapisany w kodzie URL rozumianym przez Flasha (na przykład `echoVars.swf?variable01=stuff &variable02=nachos...`).

Zapewne nie od razu zrozumiesz, że ten bardzo prosty skrypt ma ogromne możliwości. Poniższa lista zawiera tylko niektóre możliwe sposoby wykorzystania ośmiolinijkowego kodu.

- ✧ Wyświetlanie różnych filmów Flasha w zależności od danych uzyskanych od użytkownika.
- ✧ Dynamiczne przypisywanie wartości zmiennym na poziomie `_root` filmu Flasha, wykorzystując ciąg znaków w kodzie URL znajdujący się znaczniku `<OBJECT>`. Nie musisz używać niekompatybilnego języka JavaScript, w którym możesz zrobić wiele błędów.
- ✧ Obróbka danych z flashowego formularza bez używania akcji `loadVariables`.

Chociaż ten przykład jest bardzo prosty, pokazuje, że języka PHP warto używać nawet w bardzo prostych zadaniach. Możliwe jest napisanie identycznie działającego skryptu w języku JavaScript wykonującego się w przeglądarce klienta. Wymaga to jednak dużych umiejętności i zwracania uwagi na szczegóły. Na popularnych listach dyskusyjnych jest to niekończący się temat. Język JavaScript nie jest najlepszym rozwiązaniem nie tylko z powodu różnic w jego implementacji w różnych przeglądarkach, ale przede wszystkim dlatego, że nie wszystkie

przeglądarki go obsługują — strona będzie działa tylko na pewnym (przeważnie wysokim) procencie przeglądarek. Jeśli jednak skorzystasz z prostego, niewielkiego skryptu serwerowego, strona będzie działać u wszystkich użytkowników. A to dlatego, że uniezależnisz Twoją stronę od technologii używanej przez użytkownika.



Wskazówka

Na witrynach *PHP.net* i *Zend.com* znajdziesz dokumentację języka PHP. Witryny te są wystarczająco szybkie — dzięki PHP — więc możesz ich używać jako oficjalnych podręczników. Większość funkcji w języku PHP ma swoje funkcje alternatywne, które działają trochę inaczej (czasem nawet więcej niż trochę). Funkcja `include()` jest jedną z takich „zdublowanych" funkcji. Spróbuj poszukać na obydwu witrynach opisu funkcji `include()`. Znajdź także trzy inne podobne funkcje i przeczytaj o różnicach pomiędzy nimi.

Wyszukiwanie wzorca w ciągu znaków

PHP jest językiem, w którym można zrobić bardzo wiele, pisząc tylko kilka linii kodu. Wyszukiwanie wzorca jest tego dowodem. Pojedyncza linia w języku PHP używająca wyrażeń regularnych do znalezienia wzorca w ciągu znaków może robić to samo, co skrypt ActionScript liczący sobie kilkaset linii.

Wyrażenia regularne

Wyrażenie regularne to nic innego jak wzorzec wyszukiwany w ciągu znaków. Może być to nawet pojedyncza litera, gdy chcemy sprawdzić, czy właśnie ją zawiera ciąg znaków.

Przewaga wyrażeń regularnych nad prostymi metodami szukającymi ciągów znaków (jak metoda `indexOf()`) polega na tym, że możesz szukać ciągu znaków nie do końca określonego. Na przykład możesz użyć znaku wieloznacznego, który zastępuje każdy znak, podać grupę znaków (zawartych w nawiasach kwadratowych), do której musi należeć poszukiwany lub szukać wyrażenia, które musi być poprzedzone „białą" spacją. To tylko niektóre przykłady użycia wyrażeń regularnych.

Jeśli byłbyś bardzo uparty, mógłbyś napisać w języku ActionScript Flasha funkcje działające jak wyrażenia regularne, ale zajęłyby one co najmniej kilkadziesiąt linii. Pisanie takich funkcji jest, z punktu widzenia programisty, stratą czasu (nie wspominając o tym, że jest to bardzo nudne zadanie). Po co masz spędzić całe popołudnie na tworzeniu kilkusetlinijkowej funkcji, skoro to samo możesz wykonać w języku PHP, pisząc tylko dziesięć znaków.

Poszukiwanie dokładnego wzorca

Sprawdzanie istnienia wzorca w ciągu znaków jest częstym zadaniem programistycznym. W języku PHP operacje te wykonuje się bardzo łatwo. Rozważmy poniższy przykład:

```
if (ereg("PSWoods", $zmienna)){...
```

Funkcja zwróci wartość `true` (prawda), jeśli zmienna `$zmienna` zawiera tekst „PSWoods" lub „PSWoods, Esq.". Zwróci wartość `false` (fałsz), jeśli nie znajdzie podanego ciągu znaków, na przykład zmienna zawierała tekst „pswoods". Istnieje też inna funkcja, która nie bierze pod uwagę wielkości liter:

```
if (eregi("PSWoods", $zmienna)){...
```

Zwróci ona wartość `true`, jeśli zmienna `$zmienna` zawiera podany ciąg znaków („PSWoods") niezależnie od wielkości liter — na przykład „psWoods", „pswoodS" lub „pSwOoDs".

Początek i koniec ciągu znaków

Wyrażenia regularne języka PHP zawierają modyfikatory pozwalające obsłużyć różne przypadki, które chciałbyś przetestować. Na przykład chcesz sprawdzić, czy określona litera lub wzorzec znajduje się na początku testowanego tekstu:

```
if (eregi("^Dawno temu", $tekstHistoryjki)){...
```

Funkcja zwróci wartość `true`, jeśli tekst zaczyna się tak: „Dawno temu były sobie trzy misie...", ale zwróci `false`, gdy tekst jest następujący: „Były sobie trzy misie dawno temu...". Znak ^ oznacza, że poszukiwany wzorzec musi się znajdować na początku tekstu.

Następne wyrażenie zwraca `true`, jeśli na końcu tekst znajduje się ciąg znaków „Koniec.".

```
if (eregi("Koniec\.$", $tekstHistoryjki)){...
```

Znak dolara (`$`) to specjalny oznak oznaczający, że poszukiwany wzorzec musi się znajdować na końcu tekstu w zmiennej `$tekstHistoryjki`. Lewy ukośnik przed znakiem kropki jest wymagany, ponieważ znak kropki ma w przypadku wzorca specjalne znaczenie, o czym za chwilę się przekonasz. A co zrobić, jeśli chcesz, by tekst mógł kończyć się dowolnym znakiem interpunkcyjnym? Na przykład chcesz, aby funkcja zwróciła wartość `true` także w następujących przypadkach: „Koniec!", „Koniec?" lub „~Koniec~".

Symbole zastępcze

Aby możliwe było wykrycie tekstu „Koniec" zakończonego dowolnym znakiem interpunkcyjnym, używamy symbolu zastępującego każdy znak, który można umieścić w miejscu symbolu:

```
if (eregi("Koniec.$", $tekstHistoryjki)){...
```

Kropka to specjalny znak w języku PHP zastępujący dowolny inny znak. Teraz funkcja zwróci wartość `true` w każdym z wymienionych wcześniej przypadków.

Przypuśćmy, że chcemy sprawdzić, czy historyjka została napisana przez osobę, która lubi na końcu każdego listu e-mail umieszczać tekst „KONIEC!!!!!”.

```
if (eregi("Koniec!+$", $tekstHistoryjki)){...
```

Funkcja zwróci wartość `true`, jeśli po tekście „Koniec” (brak rozróżniania wielkości liter) znajdzie się co najmniej jeden wykrzyknik. Jak zapewne się domyślasz nie jest to idealne rozwiązanie, ponieważ pisarz może inaczej zakończyć tekst. Wyrażenie regularne podane w takiej formie nie obejmuje wielokrotnego powtórzenia dowolnego znaku interpunkcyjnego na końcu tekstu w zmiennej `$tekstHistoryjki` — na przykład „Koniec???”. Zapewne najbardziej oczywiste wyda się następujące określenie wzorca:

```
if (eregi("Koniec.*$", $tekstHistoryjki)){...
```

Powinieneś znać już wszystkie oznaczenia poza znakiem specjalnym *. Oznacza to samo co znak +, ale poprzedzający ten symbol znak nie musi wystąpić. Jeśli jednak zastanowisz się chwilę nad zdefiniowanym wzorcem, zauważysz, że nie musi on działać prawidłowo, ponieważ po ciągu znaków „Koniec” (bez rozróżniania wielkości liter) dowolny znak może wystąpić dowolną ilość razy. Może nie ma to znaczenia w przypadku historyjki, bo wątpię, aby ktoś zakończył ją tekstem „Koniecxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx”, ale określony wcześnie wzorzec będzie pasował także do takiego tekstu. Jesteśmy już gotowi do zapoznania się z następnym narzędziem z przybornika wyrażeń regularnych.

Grupowanie znaków

Możesz użyć nawiasu kwadratowego do zgrupowania zestawu znaków, które mogą się pojawić w danym miejscu tekstu. W tym momencie powinieneś już zauważyć, jak ogromne możliwości daje wykorzystanie wyrażeń regularnych.

```
if (eregi("Koniec[?!.,;:]*$", $tekstHistoryjki)){...
```

Nareszcie otrzymaliśmy dokładnie to, o co nam chodziło. Sprawdzamy, czy tekst w zmiennej `$tekstHistoryjki` kończy się ciągiem znaków „Koniec” (bez rozróżniania wielkości liter), za którym znajduje się zero lub więcej powtórzeń dowolnego znaku interpunkcyjnego zawartego w nawiasach kwadratowych. Jest to bardzo przydatna sztuczka. Przypuśćmy, że chcesz sprawdzić, czy w tekście znajduje się słowo „góra”, ale nie wiesz, czy użytkownik napisał je, wykorzystując polskie znaki diakrytyczne, czy też nie. W takim przypadku poszukiwany wzorzec możesz określić następująco :

```
if(eregi("g[oó]ra", $tekstHistoryjki)){...
```

Zakresy liter lub liczb można określać, stosując skróty. Na przykład poniżej sprawdzamy, czy wartość zmiennej `$nazwisko` zaczyna się od litery z pierwszej połowy alfabetu.

```
if(eregi("^[a-m]+", $nazwisko)){...
```

Możesz także wewnątrz nawiasów kwadratowych określić, które znaki *nie mogą* wystąpić, aby wzorzec się zgadzał. Gdy wewnątrz nawiasu kwadratowego użyjesz znaku ^, oznacza to, że *żaden ze znajdujących się* w nawiasie znaków nie może wystąpić na danej pozycji. Na przykład, jeśli chcesz poszukać wyrazów „daty”, „raty”, „maty”, ale nie „baty”, możesz określić wzorzec w następujący sposób:

```
if(eregi("[^b]aty", $tekstHistoryjki)){...
```

Klasy znaków

Nadal mamy problem z wzorcami, które sprawdzają, czy wartość zmiennej `$tekstHistoryjki` kończy się pewnym wariantem tekstu „Koniec", po którym dowolny znak interpunkcyjny występuje jakąś ilość razy. W poprzednim przykładzie nie wpisałem *wszystkich* znaków interpunkcyjnych, a przecież ktoś może napisać „Koniec'''".

Rozwiązaniem są klasy znaków. Klasy znaków to predefiniowane zestawy grup znaków. Na przykład klasa znaków interpunkcyjnych wygląda następująco:

```
if(eregi("Koniec[[:punct:]]*$", $tekstHistoryjki)){...
```

Poniższa tabela zawiera najczęściej używane klasy znaków, znając je, nie będziesz musiał tyle pisać.

Klasa znaków	Opis
`[:digit:]`	cyfry
`[:space:]`	dowolny znak będący „białą" spacją
`[:blank:]`	spacja lub znak tabulacji
`[:alnum:]`	znaki alfanumeryczne
`[:alpha:]`	litery, nie ma znaczenia ich wielkość
`[:lower:]`	małe litery
`[:upper:]`	wielkie litery

Przyjrzyj się poniższej linii kodu. To wyrażenie regularne sprawdza, czy zmienna `$nazwaProduktu` zawiera tekst „PriceMax" lub „PriceMax's". Regularne wyrażenie zawiera dwa nowe elementy. Pierwszy, nawiasy okrągłe, grupuje apostrof i znak „s", tak jakby były one jednym komponentem wyrażenia. Drugi, znak zapytania, oznacza, że tekst nie musi zawierać końcówki „'s". Każdy z przypadków spowoduje zwrócenie wartości `true` (prawda).

```
if(eregi("^PriceMax(`s)?$", $nazwaProduktu)){...
```

Zamiana ciągów znaków



Plik *searchResults.php* w katalogu *Rozdzial_5* na CD-ROM-ie.

Zastępowanie wzorca innym ciągiem znaków jest tak samo proste jak szukanie złożonego wzorca w tekście. Na przykład, jeśli chcesz zamienić niezbyt przyjemny wyraz „świr" z wszystkich tekstów wpisywanych do księgi gości i zastąpić go gwiazdkami, możesz użyć następującego kodu:

```
$tekstUzyt = ereg_replace("świr[aezóo]?[mew]?i?","ś***", $tekstUzyt);
$tekstUzyt = eregi_replace("świr[aezóo]?[mew]?i?","ś***", $tekstUzyt);
// to samo, ale nie ma znaczenia wielkość liter
```

Zmodyfikowany tekst zwracany przez funkcję wpisujemy do zmiennej po lewej stronie znaku równości. Jeśli zmienna `$tekstUzyt` zawierała tekst „On jest świrem. To totalny świr.", funkcja zwróci tekst „On jest ś***. To totalny ś***.".

Skrypt z pliku *searchResults.php* jest praktycznym przykładem zastępowania wzorca. Kod skryptu jest następujący:

```
$searchResults='<a href="http://macromedia.com">Macromedia</a>,
   twórcy Flasha.<br>  <a href="http://wdvl.com">WDVL</a> zawiera kilka
ćwiczeń dotyczących Flasha.<br>  <a href="http://suck.com">Suck</a>
            Pozwoli ci się śmiać z Flasha.';

$searchTerm="Flash";

$searchResults=eregi_replace( $searchTerm,
   "<B><I>".$searchTerm."</I></B>" , $searchResults);
echo $searchResults;
```

Przypuśćmy, że zmienna `$searchTerm` obejmuje wyraz szukany przez użytkownika, a zmienna `$searchResults` zawiera wynik zapytania bazy danych (tym tematem zajmiemy się w następnym rozdziale). W tym przykładzie porównujemy wzorzec podany w zmiennej `$searchTerm` z ciągiem znaków otrzymanym z zapytania (zmienna `$searchResults`) i dodajemy znaczniki pogrubienia i kursywy do każdego wyrazu pasującego do wzorca. Jest to często spotykane zastosowanie wyrażeń regularnych.

Sprawdzanie poprawności podanego adresu e-mail



Plik *validator.php* w katalogu *Rozdzial_5* na CD-ROM-ie.

Wydaje mi się, że jednym z największych problemów w formularzach flashowych jest sprawdzanie poprawności podanego przez użytkownika adresu e-mail. Na pewno ktoś na świecie pisze w tej chwili list na ten temat i umieszcza go na liście dyskusyjnej. Jak już wcześniej wspomniałem, wydaje mi się, że tego typu zadania nie są odpowiednie dla języka ActionScript Flasha. Przyjrzyj się poniższemu kodowi, znajdziesz go w pliku *validator.php*. Jest to jeszcze jeden skrypt z rekursywnym formularzem. Pierwsza część skryptu używa prostego jednolinijkowego wyrażenia do sprawdzenia, czy podany adres e-mail jest prawidłowy[3].

```
//jeśli zmiennej $submit została przypisana wartość (użytkownik
//wywołał formularz)...
if($userEmail){
  $emailPiece="[[:alnum:]]+";
  if (eregi("^$emailPiece([-_.]?$emailPiece)*@
$emailPiece([-_.]?$emailPiece)*$", $userEmail)){
    echo "Wygląda na prawidłowy<p>\n";
  }else{
    echo "To nie jest prawidłowy adres e-mail.<p>\n";
  }
```

[3] Autorowi chodzi o to, że funkcja sprawdza, czy podany adres e-mail ma prawidłowy format. Funkcja nie sprawdza, czy pod podany adres rzeczywiście można wysłać list e-mail — *przyp. tłum.*

```
}
//zawsze wyświetlaj formularz
  echo<<<ENDPAGE
  <form method="post" action="$PHP_SELF">
    <input type="text" name="userEmail" value="$userEmail">
    <input type="submit" value="Wyślij do PHP">
  </form>
ENDPAGE;
```

Znasz już wszystkie polecenia użyte w tym przykładzie; wytłumaczyć muszę tylko wyrażenie regularne. Dla każdego, kto jeszcze nie przywykł do wyrażeń regularnych, może ono wyglądać jak niezrozumiały zbiór znaków. Jeśli podzielisz je na kilka części, to łatwiej zrozumiesz.

Podstawowym elementem wyrażenia regularnego wydaje się być zmienna `$emailPiece`. Jest ona równa klasie znaków zawierającej wszystkie znaki alfanumeryczne. To podstawowy element adresu e-mail. Na przykład w fikcyjnym adresie e-mail *s.harlenhopp_sales39840@main street.sticksville.ok.us.widgets-r-us.com* każda jego część, która nie zawiera kreski, podkreślenia, kropki lub znaku @, mieści się w tej kategorii. Innymi słowy podstawowym elementem jest wszystko, co jest liczbą lub literą.

Zauważ, że w wyrażeniu regularnym określiliśmy dokładnie początek i koniec poszukiwanego ciągu znaków za pomocą znaków ^ i `$`. W ten sposób wykluczamy sytuację, w której użytkownik poza adresem e-mail podał coś jeszcze. Na przykład modyfikatory wychwycą błąd „mój e-mail to pswood@danube.com", w którym jakiś dowcipniś napisał całe zdanie, a nie wyłącznie adres e-mail.

Po pierwszym `$emailPiece` występuje wzorzec `([-_.]?$emailPiece)*`. W ten sposób dopuszczamy możliwość istnienia kolejnych grup znaków alfanumerycznych, o ile zostały poprzedzone kreską, podkreśleniem lub kropką. Pytajnik po grupie ze znakami interpunkcyjnymi oznacza, że podana grupa może (ale nie musi) wystąpić. Po prostu pozwalamy na istnieje jednego z trzech podanych znaków interpunkcyjnych między znakami alfanumerycznymi w pierwszej części adresu e-mail. Zauważ, że wyrażenie regularne zwróci wartość `false` (fałsz), jeśli jeden z tych znaków interpunkcyjnych znajdzie się na początku lub na końcu pierwszej części adresu e-mail (przed znakiem @).

Następnie występuje znak @ i powtórzenie całego wzorca sprzed znaku @. Mógłbyś usunąć znak podkreślenia z tej części (z grupy `[-_.]?`), ponieważ nazwy domen nie mogą zawierać tego znaku.

Podany przykład nie oznacza, że język PHP przydaje się tylko do sprawdzania poprawności wpisanego adresu e-mail. Jeśli jesteś zainteresowany sprawdzaniem poprawności całych formularzy, język PHP zawiera wiele przydanych narzędzi; dostępne są nawet osobne moduły. Chciałem w tym przykładzie pokazać, że do napisania całkiem dobrego programu weryfikującego poprawność adresu e-mail w języku PHP wystarczy jedna linia. Podstawową słabością podanego przykładu jest to, że nie sprawdzamy, czy użytkownik w adresie e-mail podał domenę z najwyższego poziomu (com, net, org, itp.)[4]. Nieźle, jak na jedną linię kodu.

[4] W dzisiejszych czasach sprawdzanie takich domen jest niezbyt bezpieczne, ponieważ możemy odrzucić prawidłowy adres e-mail, który nie należy do jednej z podanych przez nas grup domen — *przyp. tłum.*

Język PHP obsługuje także wzorce stosowane w języku Perl, które są bardzo podobne do tych z PHP, ale jako ograniczniki wyrażeń regularnych są stosowane ukośniki. Łącząc ten rodzaj wyrażeń regularnych z omówionym przed chwilą, możesz tworzyć jeszcze elastyczniejsze wzorce, wykorzystując granice słów, początki i końce linii w plikach, itp.

Jeśli chcesz w pełni poznać możliwości drzemiące w wyrażeniach regularnych, przeczytaj książkę na temat języka Perl[5]. Większość zagadnień związanych z wyrażeniami regularnymi opisywanych w takiej książce będziesz mógł wykorzystać nie tylko w języku Perl, ale także w językach PHP i JavaScript.

Obsługa plików


Plik *mailingList.php* z katalogu *Rozdzial_5* na CD-ROM-ie.

Istnieje naprawdę wiele zadań, które możesz wykonać, wykorzystując niewielki plik tekstowy na serwerze. Na przykład możesz wykorzystać język PHP do utworzenia niewielkich źródeł danych, skryptów dla innych programów (programu Swift Generator lub personalizacji ustawień). Istnieją jednak dwa poważne ograniczenia w stosowaniu tych plików.

Największe ryzyko wiąże się z wykorzystywaniem w języku PHP zapisu do pliku, ponieważ może się zdarzyć, że dwie osoby będą w tym samym czasie zapisywały dane do jednego pliku. Innymi słowy, dwie osoby mogą wysłać formularz do tego samego skryptu *dokładnie* w tej samej chwili, co spowoduje, że w pliku znajdą się wymieszane dane z obydwu formularzy. Można tymczasowo zablokować zapis do pliku poleceniem `flock()`, ale ta funkcja nie jest zalecana, ponieważ najnowsze wersje języka PHP (szczególnie te zainstalowane na komercyjnych serwerach) mogą działać wielowątkowo. Dlatego też tę technikę najlepiej pozostawić do zadań, w których nie ma możliwości, aby dwóch użytkowników działało jednocześnie na tym samym pliku. Są to zwykle sytuacje, w których skryptu używa tylko jedna osoba (na przykład administrator witryny flashowej używającej plików tekstowych jako źródeł danych) lub dla każdego użytkownika tworzony jest osobny plik.

Drugie ograniczenie to wydajność, choć nie ma to aż takiego znaczenia w przedstawianym przykładzie. Zapisywanie danych do pliku jest wielokrotnie wolniejsze niż zapisywanie ich do bazy danych, ponieważ większość serwerów baz danych przechowuje zestaw najczęściej wykorzystywanych danych w pamięci operacyjnej, a zapis do pliku jest ograniczony prędkością zapisu danych na dysku twardym.

Poniższy fragment kodu znajdziesz w skrypcie z pliku *mailingList.php*. (Poniższy kod jest uproszczony). Skrypt korzysta z katalogu *files*, który musi się znajdować w tym samym folderze, w którym wykonujesz skrypt.

[5] Wydawnictwo Helion (www.helion.pl) wydało kilka książek na temat języka Perl. Jedną z nich jest książka „Perl. Czarna księga" autorstwa Stevena Holznera — *przyp. tłum.*

```
$fp = fopen ("files/list.txt", "a+" );
$existingList = fread($fp,10000000);
fputs ( $fp, $userEmail."|" );
fclose ( $fp );
```

Oto typowy sposób obsługi zewnętrznych danych w języku PHP: tworzysz uchwyt pliku, który odpowiada za połączenie z zewnętrznym źródłem — w tym przypadku jest to zmienna `$fp` (nazwa uchwytu pliku jest dowolna) — manipulujesz danymi, wykorzystując ten uchwyt, a następnie zamykasz połączenie — w tym przypadku plik.

Funkcja `fopen()` jest najbardziej elastyczną z użytych powyżej funkcji. Jej pierwszym argumentem jest ścieżka i nazwa otwieranego pliku, a drugim modyfikator określający tryb otwierania pliku. Możliwe tryby wykonania tego zadania przedstawiam poniżej. Listę oparłem na podręczniku do języka PHP z witryny *Zend.com*.

- `r` — tylko do odczytu. Wskaźnik położenia w pliku domyślnie znajduje się na początku pliku.
- `r+` — odczyt i zapis. Wskaźnik położenia w pliku domyślnie znajduje się na początku pliku.
- `w` — tylko do zapisu. Wskaźnik położenia w pliku domyślnie znajduje się na początku pliku. Jeśli plik istnieje, funkcja usuwa z niego wszystkie dane. W przeciwnym przypadku tworzy nowy plik.
- `w+` — odczyt i zapis. Wskaźnik położenia w pliku domyślnie znajduje się na początku pliku. Jeśli plik istnieje, funkcja usuwa z niego wszystkie dane. W przeciwnym przypadku tworzy nowy plik.
- `a` — tylko do zapisu. Wskaźnik położenia w pliku domyślnie znajduje się na końcu pliku. Jeśli plik o podanej nazwie nie istnieje, jest tworzony.
- `a+` — odczyt i zapis. Wskaźnik położenia w pliku domyślnie znajduje się na końcu pliku. Jeśli plik o podanej nazwie nie istnieje, jest tworzony.

Funkcja `fread()` używa uchwytu pliku `$fp` utworzonego w pierwszej linii i odczytuje liczbę bajtów podaną w drugim argumencie. Jeśli funkcja dojdzie do końca pliku, ale nie odczyta zadeklarowanej liczby bajtów, zakończy działanie, nie generując błędu.

Funkcja `fputs()` zapisuje nowe dane do pliku. W powyższym przykładzie funkcja zapisze do pliku zawartość zmiennej `$userEmail` oraz znak „|”, którego używamy do oddzielania kolejnych adresów e-mail. Znak oddzielający nie ma w tym przykładzie jakiegoś szczególnego znaczenia; umieszczamy go, ponieważ musimy wiedzieć, gdzie kończy się jeden adres i zaczyna drugi, gdy zapisujemy dane w jednej linii w pliku tekstowym.

Ostatnia funkcja, `fclose()`, zamyka plik. Nowe informacje zostaną zapisane do pliku, a dane odczytane z pliku — w tym przypadku zmienna `$existingList` — pozostaną w pamięci, aby można ich było użyć w innym fragmencie skryptu.

Łączymy wszystko razem — zapisywanie się na listę e-mailową



Pliki *mailingList-SWF.php*, *mailingList.php*, *list.txt*, *subscribe.html* i *subscribe.fla* z katalogu *Rozdzial_5* na CD-ROM-ie.

Ostatnim przykładem z tego rozdziału jest plik *mailingList-SWF.php*. Użyjemy wszystkiego, co do tej pory omówiliśmy oraz długo oczekiwanej akcji `loadVariables` w filmie Flasha. W skrypcie tym dodajemy adres poczty elektronicznej użytkownika do listy e-mailowej. Skrypt pobiera adres i dodaje go do poprzednich adresów oddzielonych znakiem „|" w pliku tekstowym. Zanim zajrzymy do skryptu ActionScript z filmu Flasha, przyjrzymy się skryptowi z pliku *mailingList.php*, w którym nie korzystamy z filmu Flasha.

Powinieneś zrozumieć cały kod skryptu, ale i tak opisałem pokrótce każdy z jego fragmentów.

```
$emailPiece="[[:alnum:]]+";
if (eregi("^$emailPiece([-_.]?$emailPiece)*@$emailPiece([-
  _.]?$emailPiece)*$", $userEmail)) {
//otwarcie pliku, sprawdzenie powtarzania się adresu i zapis listy
  $fp = fopen ("files/list.txt", "a+" );
  $existingList = fread($fp,10000000);
  //sprawdzenie, czy użytkownik nie znajduje się już na liście
  if (eregi($userEmail,$existingList)){
    $errorMsg="Jesteś już zapisany.";
  //jeśli nie ma go na liście, dopisz go do pliku
  }else{
    fputs ( $fp, $userEmail."|" );
    $errorMsg="Zostałeś zapisany na listę.\n";
  }
  fclose ( $fp );
  echo $subAction;
```

Dwie najgorsze rzeczy, jakie mogą się zdarzyć podczas obsługi danych z listy e-mailowej, to podanie nieprawidłowego adresu e-mail lub jego zdublowanie na liście. Skrypt korzysta z prostych wyrażeń regularnych, aby wychwycić tego rodzaju błędy. Pierwszy fragment kodu skryptu jest tym samym wyrażeniem regularnym użytym wcześniej.

Po sprawdzeniu poprawności adresu e-mail otwieramy plik z listą i wczytujemy zawartość listy do zmiennej `$existingList`. Używamy kolejnego wyrażenia regularnego do sprawdzenia, czy wpisany przez użytkownika adres e-mail (zmienna `$userEmail`) znajduje się już na liście (zmienna `$existingList`). Jeśli tak, przypisujemy zmiennej `$errorMsg` odpowiednią wartość tekstową.

Jeśli nie znaleziono podanego adresu — instrukcja `else` struktury warunkowej — na końcu pliku zapisujemy adres e-mail oraz znak rozdzielający i informujemy użytkownika o zapisaniu na listę (modyfikacja wartości zmiennej `$errorMsg`). Następnie skrypt opuszcza strukturę warunkową i zamyka plik niezależnie od tego, czy dodał coś do pliku, czy też nie.

Sprawdź działanie skryptu w przeglądarce. Aby zobaczyć, czy skrypt rzeczywiście działa, otwórz plik *list.txt* z katalogu *files* i sprawdź, czy znajduje się tam wpisany przez Ciebie adres e-mail. Teraz wypróbuj wersję z filmem Flasha, plik *subscribe.html*. Film Flasha stanowi tylko interfejs dla tego samego skryptu — steruje formularzem i wypisywaniem wartości zmiennej `$errorMsg`. Skrypt dla flashowej wersji strony znajduje się w pliku *mailingList-SWF.php*.

Jeśli zajrzysz do pliku *subscribe.fla*, zapewne się rozczarujesz. Zawiera on tylko krótki, prosty skrypt ActionScript:

```
on (release, releaseOutside, keyPress "<Enter>"){
  loadVariables("mailingLinst-SWF.php", this, "GET");
}
```

Kod jest przypisany do przycisku wewnątrz klonu klipu filmowego `hub`. Gdy użytkownik wyśle formularz, odtwarzacz filmów Flasha zatroszczy się o to, aby zmienna `userEmail` znalazła się w adresie URL, podobnie jak w formularzu HTML wysyłanym metodą GET. Możemy w tym przykładzie skorzystać z metody POST, ale wtedy niepotrzebnie wysyłalibyśmy dodatkowe dane.

Poza tym skryptem w filmie nie ma nic ciekawego. No może tylko to, że dynamiczne pole tekstowe wyświetla wartość zmiennej `errorMsg` po wykonaniu skryptu. Jeśli masz zamiar skorzystać z podobnego sposobu w profesjonalnym filmie Flasha, powinieneś użyć licznika czasu, który po upływie określonego czasu przypisze wartość zmiennej `errorMsg` (w filmie Flasha) — jest to zabezpieczenie na wypadek, gdyby skrypt nie działał.

Co powinieneś zapamiętać z tego rozdziału

Jeśli kiedykolwiek z powodzeniem używałeś scyzoryka, narzędzia „wszystko w jednym” lub elektronicznego organizatora PDA, to zapewne przekonasz się, że jednym z takich poręcznych narzędzi w witrynach flashowych może być język PHP. Nie musisz się tym językiem specjalnie zajmować, aby go efektywnie używać. Wystarczy, że na jego naukę poświęcisz kilka wieczorów, a oszczędzisz sobie mnóstwa pracy w przyszłości. Będziesz mógł pisać w języku PHP zadania trudne do zaprogramowania w języku ActionScript i skupisz się na wykorzystywaniu zalet filmów Flasha. Innymi słowy, Twoje aplikacje flashowe będą coraz lepsze.