PHP. Receptury. Wydanie II

Autorzy: Adam Trachtenberg, David Sklar
Tłumaczenie: Marek Pałczyński
ISBN: 978-83-246-0827-0
Tytuł oryginału: PHP Cookbook
Format: B5, stron: 816
Przykłady na ftp: 194 kB

Podręczny słownik 250 sprawdzonych i skutecznych rozwiązań z zakresu języka PHP

- Praca z typami danych i blokami programów PHP
- Obsługa formularzy, baz danych i sesji
- Tworzenie wydajnych i bezpiecznych witryn internetowych

PHP to najpopularniejszy język skryptowy na świecie, wykorzystywany w milionach witryny internetowych. Szeroki wachlarz możliwości, łatwa składnia oraz współpraca z wieloma systemami operacyjnymi sprawiają, że jest to idealne narzędzie do tworzenia dynamicznych aplikacji WWW. Od wersji PHP 5 język ten obsługuje programowanie obiektowe oraz udostępnia usprawniony mechanizm współpracy z bazami danych, co w znacznym stopniu ułatwia kreowanie rozbudowanych programów sieciowych.

Drugie wydanie książki „PHP. Receptury" zawiera jeszcze więcej gotowych rozwiązań, które zostały dostosowane do nowej wersji języka, czyli PHP 5. Dodatkowo ulepszona forma umożliwia łatwiejsze znalezienie potrzebnych receptur. Pozwolą Ci one szybko rozwiązać często spotykane problemy. Czytając tę książkę dowiesz się między innymi, jak wykonywać operacje na różnych typach danych, jakie elementy składają się na programy PHP i jak z nich korzystać, a także jak obsługiwać formularze czy współpracować z bazami danych. Nauczysz się stosować techniki zarządzania sesjami. Poznasz zasady korzystania z XML, współpracy z kodem w języku JavaScript, a także rozwiązania wielu innych praktycznych problemów.

- Praca z typami danych
- Korzystanie z różnych bloków aplikacji PHP
- Obsługa formularzy
- Praca z bazami danych
- Przetwarzanie dokumentów XML
- Usługi Web Services
- Zarządzanie sesją
- Generowanie grafiki na stronach internetowych
- Zabezpieczanie witryn
- Obsługa błędów
- Optymalizacja kodu
- Praca z systemem plików i katalogów

Wykorzystaj gotowy kod do błyskawicznego tworzenia dynamicznych witryn internetowych

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

ROZDZIAŁ 8.

Podstawy programowania na potrzeby WWW

8.0. Wprowadzenie

Poszukiwanie informacji na temat programowania dla WWW to zapewne jedna z głównych przyczyn, dla których Czytelnik sięga po tę książkę. Z kolei potrzeba programowania dla WWW była jedną z najważniejszych przyczyn powstania samego PHP oraz uczyniła ten język tak popularnym. Budowanie dynamicznych, niemal nieograniczonych w swoich możliwościach programów WWW za pomocą PHP nie jest skomplikowanym zadaniem. W innych rozdziałach książki zostały omówione różnorodne funkcje języka, takie jak obsługa grafiki, wyrażeń regularnych, dostępu do baz danych i operowania plikami. Wszystkie one stanowią elementy ogólnie pojętego programowania WWW. W niniejszym rozdziale skoncentrujemy się jednak tylko na pewnych problemach charakterystycznych dla Internetu oraz na zagadnieniach organizacyjnych, które pomogą usprawnić proces programowania.

Receptury 8.1, 8.2 i 8.3 demonstrują sposób zapisywania, odczytywania i usuwania danych cookie. Cookie to niewielkich rozmiarów ciąg tekstowy, który na polecenie serwera przeglądarka przesyła wraz z żądaniem strony. Gromadzenie danych cookie w przeglądarce jest efektem działania skryptów dołączanych do stron przeglądanej witryny. Żądania HTTP z założenia są „bezstanowe", tzn. dane żądanie nie może być wiązane z poprzednim. Tymczasem użycie danych cookie pozwala na łączenie różnych żądań wygenerowanych przez tego samego użytkownika. Własność ta ułatwia obsługę „koszyków" w sklepach internetowych oraz umożliwia rejestrowanie realizowanych przez użytkownika wyszukiwań.

Receptura 8.4 przedstawia sposób odsyłania użytkownika do innej strony WWW. Z kolei receptura 8.5 wyjaśnia zagadnienia związane z pozyskiwaniem informacji o przeglądarce klienta. Receptura 8.6 zajmuje się problemem konstruowania adresu URL uzupełnionego zapytaniem metody GET, uwzględniając zagadnienie właściwego kodowania znaków specjalnych oraz obsługę elementów HTML. W recepturze 8.7 zostały natomiast zamieszczone informacje na temat pobierania danych przekazanych przez użytkownika za pomocą żądań POST. Receptura 8.8 zawiera rozwiązanie problemu, który często występuje podczas formatowania danych WWW — wyświetlanie wierszy tabeli HTML w różnych kolorach i z wykorzystaniem różnych atrybutów stylu.

Dwa kolejne podrozdziały prezentują zasady korzystania z mechanizmu uwierzytelniania, umożliwiającego ochronę stron WWW za pomocą haseł. Funkcje PHP związane z podstawowym uwierzytelnianiem HTTP są przedstawione w recepturze 8.9. Natomiast receptura 8.10 zilustruje inną metodę, polegającą na wykorzystaniu danych cookies, co niekiedy okazuje się korzystniejszym rozwiązaniem.

Trzy następne receptury dotyczą sterowania danymi wyjściowymi. Podrozdział 8.11 prezentuje mechanizm wymuszonego przesyłania danych do przeglądarki. Receptura 8.12 przedstawia funkcje buforowania informacji wyjściowych, przydatne w sytuacjach, gdy konieczne jest zgromadzenie danych przeznaczonych do wydruku lub opóźnienie przesłania treści do momentu przetworzenia kodu całej strony. Z kolei w podrozdziale 8.13 omówiono zagadnienie automatycznej kompresji danych wyjściowych.

Kolejne dwie receptury zawierają opis sposobu korzystania z zewnętrznych zmiennych — zmiennych środowiskowych i ustawień konfiguracyjnych interpretera PHP. Zagadnienia te zostały przedstawione w recepturach 8.14 i 8.15. Informacje zamieszczone w recepturze 8.16 są szczególnie istotne dla osób, które korzystają z serwera WWW Apache. Dotyczą bowiem komunikacji programów PHP z różnymi modułami Apache.

W końcowej części rozdziału zostały również zaprezentowane przykłady programów, które uwzględniają w działaniu większość opisywanych tu rozwiązań. Zadanie programu 8.17 polega na zatwierdzaniu kont użytkowników przez wysyłanie listów elektronicznych zawierających w treści charakterystyczny dla danego użytkownika odsyłacz. Jeżeli w ciągu tygodnia użytkownik nie otworzy za jego pomocą przygotowanej strony internetowej, konto zostanie usunięte. Natomiast program 8.18 jest przykładem nieskomplikowanego serwisu Wiki — systemu, który umożliwia edytowanie dowolnych stron witryny za pomocą przeglądarki internetowej.

8.1. Zapisywanie danych cookie

Problem

Chcemy zapisać dane cookie, aby aplikacja WWW mogła rozpoznawać kolejne żądania generowane przez tę samą przeglądarkę internetową.

Rozwiązanie

Należy wywołać funkcję `setcookie()`, podając nazwę i wartość pola, tak jak to zostało przedstawione w listingu 8.1.

Listing 8.1. Zapisywanie danych cookie

```
<?php
setcookie('smak', 'czekoladowy');
?>
```

Analiza

Dane cookies przysyłane są w ramach nagłówka HTTP. Z tego względu funkcja `setcookie()` musi być wywoływana przed wygenerowaniem jakiejkolwiek treści strony.

Dopuszczalne jest przekazywanie do funkcji `setcookie()` dodatkowych parametrów, które umożliwiają sterowanie funkcjonowaniem cookie. Trzecim możliwym argumentem wywołania `setcookie()` jest czas wygaśnięcia danych, wyrażony w postaci znacznika czasowego. Przykładowo, dane cookie, które będą przechowywane do 3. grudnia 2004 do południa czasu GMT, zapisywane są przy użyciu funkcji przedstawionej w listingu 8.2.

Listing 8.2. Definiowanie wygasających danych cookie

```
setcookie('smak', 'czekoladowy', 1102075200);
```

W przypadku, gdy trzeci z argumentów nie został określony (lub gdy jest pusty), dane cookie są usuwane wraz z zakończeniem pracy przeglądarki. W wielu systemach znacznik czasowy jest ograniczony do wartości 2147483647, gdyż jest to maksymalna wartość całkowitoliczbowa, jaką można zapisać przy wykorzystaniu znacznika 32-bitowego (o czym informowaliśmy we wprowadzeniu do rozdziału 3.).

Czwarty argument `setcookie()` to ścieżka. Dane cookie są odsyłane do serwera tylko w przypadku, gdy ścieżka żądanej strony rozpoczyna się od podanej wartości tekstowej. Przykładowo, dane cookie zapisane w sposób przedstawiony w listingu 8.3 zostanę odesłane do serwera tylko wtedy, gdy ścieżka strony rozpoczyna się od ciągu */wyroby/*:

Listing 8.3. Definiowanie danych cookie z wyznaczeniem katalogu serwera

```
<?php
setcookie('smak', 'czekoladowy', '', '/wyroby/');
?>
```

Ścieżka do strony zapisującej dane cookie nie musi rozpoczynać się wartością */wyroby/*, niemniej tylko do takich stron będą dane odsyłane.

Piątym parametrem funkcji jest domena. Dane cookie są odsyłane do serwera tylko w przypadku, gdy żądane są strony, których nazwa serwera kończy się określoną nazwą domenową. W przykładach prezentowanych w listingu 8.4 dane cookie zdefiniowane pierwszą z funkcji będą odsyłane do wszystkich serwerów z domeny *przyklad.com*, a te zapisane za pomocą drugiej funkcji — jedynie do komputera *joanna.przyklad.com*:

Listing 8.4. Definiowanie danych cookie z wyznaczeniem domeny

```
<?php
setcookie('smak', 'czekoladowy', '', '', '.przyklad.com');
setcookie('smak', 'czekoladowy', '', '', '.joanna.przyklad.com');
?>
```

Gdyby w pierwszym przypadku jako wartości domeny użyto tylko *przyklad.com* zamiast *.przyklad.com*, dane mogłyby być przesyłane jedynie do pojedynczego komputera o nazwie *przyklad.com* (a nie na przykład do *www.przyklad.com* czy *joanna.przyklad.com*).

Ostatnim (opcjonalnym) argumentem wywołania funkcji jest znacznik, który ustawiony na 1 informuje przeglądarkę o konieczności przesyłania danych cookie tylko w ramach połączenia korzystającego z protokołu SSL. Właściwość ta bywa użyteczna w przypadkach, gdy wymieniane informacje są szczególnie cenne. Musimy jednak pamiętać, że dane cookie są przechowywane w komputerze użytkownika w postaci niezaszyfrowanej.

Poszczególne przeglądarki mogą traktować cookie w nieco odmienny sposób. Rzecz dotyczy w szczególności ustalania stopnia zgodności z daną ścieżką i domeną oraz priorytetów pomiędzy różnymi danymi cookie o tej samej nazwie. Szczegółowe wyjaśnienie wspomnianych różnic zamieszczono na poświęconej funkcji `setcookie()` internetowej stronie podręcznika.

Zobacz również

Sposób odczytywania danych cookie przedstawiono w recepturze 8.2. Podrozdział 8.3 omawia zagadnienie ich usuwania. Receptura 8.12 wyjaśnia koncepcję buforowania danych. Dokumentacja `setcookie()` znajduje się pod adresem *http://www. php.net/setcookie*, a szczegółowa specyfikacja idei danych cookie — w dokumencie RFC 2965, pod adresem *http://www.faqs.org/rfcs/rfc2965.html*.

8.2. Odczytywanie danych cookie

Problem

Chcemy odczytać zapisane wcześniej dane cookie.

Rozwiązanie

Rozwiązanie polega na przeanalizowaniu zawartości tablicy globalnej `$_COOKIE` zgodnie z instrukcjami zamieszczonymi w listingu 8.5.

Listing 8.5. Odczytywanie danych cookie

```
<?php
if (isset($_COOKIE['smak'])) {
  print "Zjedzone ciasteczka miały $_COOKIE['smak'] smak.";
}
?>
```

Analiza

Wartości cookie nie są dostępne za pośrednictwem `$_COOKIE` w trakcie realizacji żądania zapisującego dane informacje cookie. Innymi słowy, funkcja `setcookie()` nie modyfikuje wartości `$_COOKIE`. Wszystkie zapisane dane cookie są dostępne dopiero w następnych żądaniach. Jeśli dyrektywa `register_globals` ma wartość `on`, dane cookie przypisywane są także zmiennym globalnym.

Odsyłając cookie do serwera, przeglądarka przesyła jedynie wartość. Nie istnieje możliwość operowania za pośrednictwem `$_COOKIE` informacjami o domenie, ścieżce, terminie wygasania czy stopniu zabezpieczenia. Przeglądarka nie przesyła takich informacji do serwera.

Aby wyświetlić nazwy i odpowiadające im wartości cookie danego żądania, możemy użyć pętli, która przeanalizuje tablicę `$_COOKIE` w sposób przedstawiony w listingu 8.6.

Listing 8.6. Odczyt wszystkich danych cookie

```
<?php
foreach ($_COOKIE as $nazwa_cookie => $wartosc_cookie) {
  print "$nazwa_cookie = $cookie_value<br>";
}
?>
```

Zobacz również

Zapis danych cookie opisano w recepturze 8.1. Podrozdział 8.3 omawia zagadnienie ich usuwania. Receptura 8.12 wyjaśnia koncepcję buforowania danych. Informacje o `register_globals` zawarto w recepturze 9.15.

8.3. Usuwanie danych cookie

Problem

Chcemy usunąć dane cookie, aby przeglądarka nie mogła ich odesłać do serwera. Taka operacja jest niezbędna do wylogowania użytkownika z systemu, który wykorzystuje dane cookie do sprawdzania, czy użytkownik jest uwierzytelniony.

Rozwiązanie

Wywołanie `setcookie()` bez żadnej wartości i z przeszłą datą wygasania pozwoli rozwiązać problem. Stosowna instrukcja został przedstawiona w listingu 8.7.

Listing 8.7. Usuwanie danych cookie

```
<?php
setcookie('smak', '', 1);
?>
```

Analiza

Właściwym rozwiązaniem problemu jest ustalenie czasu wygasania znacznie wcześniejszego niż bieżący. Zapobiega to komplikacjom w przypadku, gdyby komputer klienta i serwer nie miały zsynchronizowanych zegarów. Przykładowo, jeżeli według serwera aktualna godzina to 15:06, a zegar klienta wskazuje 15:02, cookie z czasem wygasania o wartości 15:05 nie zostanie usunięte przez komputer użytkownika, mimo iż dla serwera jest to już przeszłość.

Wywołując `setcookie()` w celu usunięcia danych cookie, należy podać dokładnie takie same parametry (poza wartością i czasem), jakie podano w `setcookie()` podczas zapisywania informacji, tzn. domenę, ścieżkę i znacznik zabezpieczenia, o ile takowe zostały określone.

Zobacz również

Zapis danych cookie opisano w recepturze 8.1. Podrozdział 8.2 omawia zagadnienie odczytu ich wartości. Receptura 8.12 wyjaśnia koncepcję buforowania danych. Dokumentacja funkcji `setcookie()` znajduje się pod adresem *http://www.php.net/setcookie*.

8.4. Odsyłanie do innej strony

Problem

Chcemy automatycznie odsyłać użytkownika do strony o innym adresie URL. Przykładem zastosowania takiego rozwiązania może być sytuacja, kiedy przekazane za pomocą formularza dane zostały zapisane i należy odesłać użytkownika do strony, która poinformuje go o poprawnym zakończeniu operacji.

Rozwiązanie

Użyjemy funkcji `header()` do przesłania nagłówka `Location` z nowym adresem, a następnie instrukcji `exit()`, uniemożliwiającej przesłanie jakiejkolwiek treści do przeglądarki. Sposób realizacji zadania został przedstawiony w listingu 8.8.

Listing 8.8. Odesłanie użytkownika do innej strony

```
<?php
header('Location: http://www.przyklad.com/');
exit();
?>
```

Analiza

Jeżeli zachodzi potrzeba przekazania do nowej strony jakichkolwiek zmiennych, możemy je umieścić w dołączonym do adresu URL ciągu tekstowym zapytania, tak jak to zostało pokazane w listingu 8.9.

Listing 8.9. Odesłanie do innej strony z uwzględnieniem zmiennych ciągu zapytania

```
<?php
header('Location: http://www.przyklad.com/?zwierze=pies');
exit();
?>
```

Docelowy ciąg URL powinien zawierać informacje o nazwie protokołu i nazwie jednostki. Nie wystarczy zapisanie jedynie ścieżki dostępu do pliku. W listingu 8.10 został przedstawiony przykład właściwego i błędnego ciągu `Location` nagłówka HTTP.

Listing 8.10. Poprawny i błędny ciąg Location

```
<?php
// Poprawne przekierowanie
header('Location: http://www.przyklad.com/katalog/zywnosc/mielonka.php');

// Błędne przekierowanie
header('Location: /katalog/zywnosc/mielonka.php');
?>
```

Nowa strona, do której użytkownik jest odsyłany, pozyskiwana jest za pomocą metody GET. Dopuszcza się również podobną operację z użyciem metody POST. Dzięki językowi JavaScript możliwe jest zasymulowanie odesłania z wykorzystaniem metody POST. Wystarczy

wygenerować formularz, który zostanie automatycznie odesłany. Gdy przeglądarka obsługująca język JavaScript pobierze stronę z listingu 8.11, natychmiast prześle za pomocą metody POST zawarty w kodzie formularz.

Listing 8.11. Odesłanie z wykorzystaniem formularza przesyłanego za pomocą metody POST

```
<html>
  <body onload="document.getElementById('redirectForm').submit()">
    <form id='redirectForm' method='POST' action='./zrobione.html'>
      <input type='hidden' name='status' value='zakonczone' />
      <input type='hidden' name='id' value='Ou812' />
      <input type='sumbit' value='Kliknij tutaj, aby kontynuować' />
    </form>
  </body>
</html>
```

Wartością atrybutu `id` formularza przedstawionego w listingu 8.11 jest ciąg `redirectForm`. Zatem instrukcja przypisana do zdarzenia `onload` elementu `<body>` spowoduje przesłanie tego formularza. Nie zostanie ona jednaj wykonana, jeśli przeglądarka nie obsługuje języka JavaScript. W takim przypadku użytkownik zobaczy na ekranie przycisk z informacją *Kliknij tutaj, aby kontynuować*.

Zobacz również

Dokumentacja funkcji `header()` jest dostępna pod adresem *http://www.php.net/header*.

8.5. Pozyskiwanie informacji o przeglądarkach

Problem

Chcemy, aby generowana treść strony zależała od właściwości danej przeglądarki.

Rozwiązanie

W tym celu wykorzystamy obiekt zwracany przez funkcję `get_browser()`, który umożliwia ustalenie parametrów przeglądarki. Sposób pobrania danych został przedstawiony w listingu 8.12.

Listing 8.12. Pobranie informacji o przeglądarce

```
<?php
$browser = get_browser( );
if ($browser->frames) {
    // treść wykorzystująca ramki
} elseif ($browser->tables) {
    // treść wykorzystująca tabele
} else {
    // klasyczna treść HTML
}
?>
```

Analiza

Działanie funkcji `get_browser()` sprowadza się do przeanalizowania zmiennej środowiskowej `$_ENV['HTTP_USER_AGENT']` (definiowanej przez serwer WWW) i porównania jej wartości z listą przeglądarek, która znajduje się w zewnętrznym pliku cech przeglądarek. Ze względu na problem licencji, PHP nie jest rozpowszechniany z plikiem cech przeglądarek. Jednak PHP FAQ w sekcji *Pozyskiwanie PHP* (ang. *Obtaining PHP*) (*http://www. php.net/faq.obtaining*) jako strony, z których można pobrać pliki o możliwościach poszczególnych przeglądarek, wymienia: *http://www.cyscape.com/asp/browscap/* oraz *http://www.amrein. com/apps/page.asp?Q=InowDownload*. Innym źródłem jest *http://asp.net.do/browscap.zip*.

Po pobraniu pliku cech przeglądarek konieczne jest zamieszczenie informacji o jego położeniu. W tym celu należy nadać dyrektywie konfiguracyjnej PHP `browscap` wartość ścieżki do wspomnianego pliku. Jeżeli PHP jest wykorzystywany jako skrypt CGI, dyrektywa powinna się znaleźć w pliku *php.ini* i mieć treść przedstawioną w listingu 8.13.

Listing 8.13. Ustawienie wartości browscap w pliku php.ini

```
browscap=/usr/local/lib/browscap.txt
```

Większość cech przeglądarek udostępnianych przez funkcję `get_browser()` zestawiono w tabeli 8.1. Funkcja umożliwia rozpoznanie, czy przeglądarka obsługuje takie elementy jak `javascript` czy dane `cookies`. Ostatecznie jednak o dostępności wspomnianych elementów decyduje użytkownik, a funkcja nie możliwości poinformowania o tym, czy klient pozwolił na ich obsługę. Funkcja `get_browser()` wskazuje na możliwość stosowania skryptów JavaScript nawet w przypadku, gdy w zestawieniu przeglądarek obsługujących JavaScript znajduje się odmienna informacja na ten temat. Podobnie, może dostarczać informacji o możliwości zapisu danych cookies, mimo iż klient odmawia wykonania takiej operacji.

Tabela 8.1. Własności obiektu reprezentującego cechy przeglądarki

Własność	Opis
`platform`	System operacyjny, w którym przeglądarka jest uruchomiona (np. Windows, Macintosh, Unix, Win32, Linux, MacPPC)
`version`	Pełny numer wersji (np. 5.0,3.5,6.0b2)
`majorver`	Zasadniczy numer wersji (np. 5,3,6)
`minorver`	Poboczny numer wersji (np. 0,5,02)
`frames`	Przechowuje wartość `1`, jeżeli przeglądarka obsługuje ramki
`tables`	Przechowuje wartość `1`, jeżeli przeglądarka obsługuje tabele
`cookies`	Przechowuje wartość `1`, jeżeli przeglądarka akceptuje dane cookies
`backgroundsounds`	Przechowuje wartość `1`, jeżeli przeglądarka pozwala na odtwarzanie muzyki za pomocą znaczników `<embed>` lub `<bgsound>`
`vbscript`	Przechowuje wartość `1`, jeżeli przeglądarka obsługuje VBScript
`javascript`	Przechowuje wartość `1`, jeżeli przeglądarka obsługuje JavaScript
`javaapplets`	Przechowuje wartość `1`, jeżeli przeglądarka umożliwia uruchamianie apletów Javy
`activexcontrols`	Przechowuje wartość `1`, jeżeli przeglądarka umożliwia uruchamianie kontrolek ActiveX

Zobacz również

Dokumentacja funkcji `get_browser()` jest dostępna pod adresem *http://www.php.net/get-browser*.

8.6. Konstruowanie zapytania metody GET

Problem

Chcemy utworzyć łącze zawierające w zapytaniu pary nazwa-wartość.

Rozwiązanie

Zadanie to realizuje funkcja `http_build_query()`, przedstawiona w listingu 8.14.

Listing 8.14. Konstruowanie zapytania metody GET

```
<?php
$vars = array('nazwisko' => 'Kermit Zaba',
              'kolor' => 'zielony',
              'znak_wypunktowania' => '#');
$query_string = http_build_query($vars);
$url = '/muppety/wybor.php?' . $query_string;
?>
```

Analiza

Adres URL, utworzony na podstawie kodu z listingu 8.14, miałby następującą postać:

```
/muppety/wybor.php?nazwisko=Kermit+Zaba&kolor=zielony&znak_wypunktowania=%23
```

W zapytaniu znajdują się spacje, które zakodowano za pomocą znaków plus (+). Znaki specjalne, takie jak (#), reprezentowane są odpowiednią wartością heksadecymalną. W tym przypadku jest to wartość `%23`, z uwagi na fakt, że kodem ASCII odpowiadającym znakowi (#) jest `35`, co w notacji szesnastkowej odpowiada wartości `23`.

Funkcja `urlencode()` zapewnia, że wszystkie znaki specjalne wchodzące w skład nazw lub wartości będą odpowiednio zapisane w adresie URL. Problem może się pojawić jedynie wtedy, gdy nazwa zmiennej rozpoczyna się ciągiem tekstowym zgodnym z wartością występującą w języku HTML. Przykładem może być część adresu URL, który przekazuje informacje o sygnale okresowym:

```
/sygnal.php?czestot=1000&amp=10
```

W języku HTML do reprezentacji znaku (`&`) jest stosowany ciąg tekstowy `&amp`. Z tego powodu przeglądarka może zinterpretować adres URL jako:

```
/sygnal.php?czestot=1000&=10
```

Istnieją trzy sposoby unikania takich sytuacji. Pierwszy z nich polega na zastosowaniu nazw zmiennych, które nie wchodziłyby w konflikt z elementami języka — na przykład `_amp` zamiast `amp`. Drugim jest przekształcenie znaków w odpowiednikach elementów HTML do postaci tych elementów języka przed wygenerowaniem adresu URL. Służy do tego instrukcja `htmlentities()`:

```
$url = '/muppety/wybor.php?' . htmlentities($query_string);
```

W wyniku otrzymujemy:

```
/muppety/wybor.php?nazwisko=Kermit+Zaba&amp;kolor=zielony&amp;znak_wypunktowania=%23
```

Trzecią możliwością jest zmiana znaku rozdzielającego i zastąpienie znaku `&` znakiem `;`. W tym celu należy ustalić wartość dyrektywy `arg_separator` na `;`. Pary nazwa-wartość są wówczas rozdzielane znakiem średnika:

```
/muppety/wybor.php?nazwisko=Kermit+Zaba;kolor=zielony;znak_wypunktowania=%23
```

Zobacz również

Omówienie funkcji `urlencode()` znajduje się pod adresem *http://www.php.net/urlencode*, a `htmlentities()` — pod adresem *http://www.php.net/htmlentities*.

8.7. Odczytywanie treści żądania POST

Problem

Niezbędny jest bezpośredni dostęp do treści żądania POST, a nie tylko do przeanalizowanych przez PHP danych udostępnianych w tablicy `$_POST`. Taki problem może wystąpić wówczas, gdy trzeba przetworzyć dokument XML przekazany do serwera w formie żądania kierowanego do usługi WWW.

Rozwiązanie

Dane należy odczytać ze strumienia `php://input` w sposób przedstawiony w listingu 8.15.

Listing 8.15. Odczyt treści żądania POST

```
<?php
$body = file_get_contents('php://input');
?>
```

Analiza

Automatycznie tworzona globalna tablica `$_POST` doskonale spełnia swoje zadanie, gdy potrzebny jest dostęp do zmiennych przesyłanego formularza. Jest jednak bezużyteczna, gdy programista musi w nieograniczony sposób operować nieprzetworzoną treścią całego żądania. Przydatny okazuje się wówczas strumień `php://input`. Pozwala on na zastosowanie funkcji `file_get_contents()` do odczytania całej treści żądania lub na wykorzystanie funkcji `fread()` do pobierania kolejnych fragmentów tego żądania.

Jeżeli dyrektywa konfiguracyjna `always_populate_raw_post_data` ma wartość `on`, nieprzetworzone dane metody POST są również umieszczane w globalnej zmiennej `$HTTP_RAW_POST_DATA`. Chcąc jednak przygotować kod o możliwie największym stopniu przenośności, warto wykorzystać rozwiązanie uwzględniające strumień `php://output` — spełnia ono bowiem swoje zadanie nawet wtedy, gdy opcja `always_populate_raw_post_data` jest wyłączona.

Zobacz również

Dokumentacja strumienia `php://input` znajduje się pod adresem *http://www.php.net/wrappers*, natomiast dyrektywy `always_populate_raw_post_data` — pod adresem *http://www.php.net/ini.core#ini.always-populate-raw-post-data*.

8.8. Tabele HTML z wierszami o różnych atrybutach stylu

Problem

Chcemy wyświetlić tabelę z danymi, w której styl prezentacji wierszy będzie różny dla różnych wierszy. Na przykład wiersze o numerach nieparzystych będą miały białe tło, a wiersze parzyste — szare tło.

Rozwiązanie

Wystarczy podczas generowania kodu HTML tabeli wykorzystywać naprzemiennie dwie klasy stylu CSS. Przykład wykorzystania tej techniki do wyświetlenia danych pozyskanych z bazy danych został przedstawiony w listingu 8.16.

Listing 8.16. Utworzenie tabeli HTML o różnych stylach wierszy

```
<style type="text/css">
.even-row {
    background: white;
}
.odd-row {
    background: gray;
}
</style>
<table>
<tr><th>Ilość</th><th>Dodatek</th></tr>
<?php
$styles = array('even-row', 'odd-row');
$db = new PDO('sqlite:altrow.db');
foreach ($db->query('SELECT quantity, ingredient FROM ingredients') as $i => $row) {
?>
<tr class="<?php echo $styles[$i % 2]; ?>">
  <td><?php echo htmlentities($row['quantity']) ?></td>
  <td><?php echo htmlentities($row['ingredients']) ?></td></tr>
<?php } ?>
</table>
```

Analiza

Zwięzłość kodu przedstawionego w listingu 8.16 wynika z zastosowania tablicy nazw klas CSS (`$styles`) i operatora „reszty ” (`%`). Operator reszty zwraca wartość reszty z dzielenia całkowitego dwóch liczb. Reszta z dzielenia dowolnej wartości przez dwa (w tym przypadku dzielony jest numer wiersza) zawsze wynosi `0` lub `1`. Dzięki temu można w łatwy sposób naprzemiennie stosować pierwszy i drugi element tablicy `$styles`.

Zobacz również

Dokumentacja operatorów arytmetycznych języka PHP jest dostępna pod adresem *http://www.php.net/language.operators.arithmetic*.

8.9. Proste uwierzytelnianie HTTP

Problem

Chcemy zabezpieczyć dostęp do pewnej części witryny WWW za pomocą haseł. Mechanizm weryfikacji haseł powinien zostać zdefiniowany w programie PHP, co pozwoli na wyeliminowanie konieczności przechowywania ich w plikach zewnętrznych i zwolni serwer WWW z obowiązku przeprowadzania uwierzytelniania.

Rozwiązanie

Jeżeli użytkownik podał swoją nazwę i hasło, dane te są przechowywane w zmiennych globalnych `$_SERVER['PHP_AUTH_USER']` i `$_SERVER['PHP_AUTH_PW']`. Aby uniemożliwić dostęp do strony, musimy przesłać nagłówek `WWW-Authenticate`, w którym powinna być zawarta również informacja o obszarze witryny objętym uwierzytelnianiem oraz kod odpowiedzi o wartości 401. Stosowny kod został przedstawiony w listingu 8.17.

Listing 8.17. Wymuszenie uwierzytelnienia typu Basic

```
<?php
header('WWW-Authenticate: Basic realm="Moja strona"');
header('HTTP/1.0 401 Unauthorized');
echo "Uzyskanie dostępu do strony wymaga podania poprawnej nazwy użytkownika i
hasła.";
exit();
?>
```

Analiza

Kiedy przeglądarka otrzyma nagłówek z kodem 401, wyświetli okno dialogowe umożliwiające wprowadzenie nazwy użytkownika i hasła. Przekazane w ten sposób parametry uwierzytelniania (nazwa i hasło) — o ile zostaną zaakceptowane przez serwer — będą związane z obszarem witryny określonym w nagłówku `WWW-Authenticate` (pole `realm`). Z uwagi na fakt, że kod, który przetwarza parametry uwierzytelniania, może generować nagłówki, jego wykonanie powinno poprzedzać przesłanie jakiejkolwiek treści strony. Można do tego celu wykorzystać na przykład funkcję `pc_validate()`, przedstawioną w listingu 8.18.

Listing 8.18. Funkcja pc_validate()

```
<?php
function pc_validate($user,$pass) {
    /* w tym miejscu należy zdefiniować odpowiedni mechanizm weryfikacji nazwy
       i hasła użytkownika, np. sprawdzenie wartości w bazie danych */
    $users = array('dawid' => 'fadj&32',
                   'adam'  => '8HEj838');
```

```
    if (isset($users[$user]) && ($users[$user] == $pass)) {
        return true;
    } else {
        return false;
    }
}
?>
```

Przykład zastosowania funkcji `pc_validate()` został zamieszczony w listingu 8.19.

Listing 8.19. Wykorzystanie funkcji weryfikacji danych uwierzytelniających

```
<?php
if (! pc_validate($_SERVER['PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW'])) {
    header('WWW-Authenticate: Basic realm="Moja strona"');
    header('HTTP/1.0 401 Unauthorized');
    echo "Uzyskanie dostępu do strony wymaga podania poprawnej nazwy
          użytkownika i hasła.";
    exit;
}
?>
```

W treści funkcji `pc_validate()` należy umieścić odpowiedni kod, który pozwoli na sprawdzenie, czy użytkownik podał poprawną nazwę i hasło. Zmodyfikować można również ciąg tekstowy identyfikujący obszar witryny objęty uwierzytelnianiem oraz komunikat wyświetlany w sytuacji, gdy użytkownik kliknie przycisk *Anuluj* w oknie uwierzytelniania (tu będzie to komunikat: *Uzyskanie dostępu do strony wymaga podania poprawnej nazwy użytkownika i hasła*.).

W PHP 5.1.0 i późniejszych wersjach języka obsługiwane jest również uwierzytelnianie typu `Digest`. W przypadku prostego uwierzytelniania (uwierzytelniania typu `Basic`) nazwy użytkowników i ich hasła są przesyłane do serwera w formie otwartego tekstu, nieznacznie tylko zabezpieczone przez algorytm kodowania Base64. W uwierzytelnianiu typu `Digest` hasło nigdy nie jest przekazywane z przeglądarki do serwera. Zamiast niego dostarczane są jedynie wartości skrótu hasła oraz kilka parametrów uzupełniających. Takie rozwiązanie zmniejsza ryzyko wykorzystania danych przez osobę, która je przechwyciła. Oczywiście zwiększony poziom bezpieczeństwa mechanizmu `Digest` oznacza również zwiększenie złożoności kodu skryptu w porównaniu z algorytmem prostego uwierzytelniania. W listingu 8.20 zostały przedstawione funkcje, które wyznaczają parametry uwierzytelniania `Digest` zgodnie z zaleceniami określonymi w dokumencie RFC 2617.

Listing 8.20. Wykorzystanie uwierzytelniania typu Digest

```
<?php

/* w tym miejscu należy zdefiniować odpowiedni mechanizm weryfikacji nazwy
   i hasła użytkownika, np. sprawdzenie wartości w bazie danych */
$users = array('dawid' => 'fadj&32',
               'adam'  => '8HEj838');
$realm = 'Moja strona';

$username = pc_validate_digest($realm, $users);

// Poniższa instrukcja nie zostanie wykonana, jeśli dane uwierzytelniające są
// niepoprawne
print "Witaj, " . htmlentities($username);

function pc_validate_digest($realm, $users) {
    // Zakończenie z błędem, jeśli klient nie dostarczył wartości skrótu
    if (! isset($_SERVER['PHP_AUTH_DIGEST'])) {
        pc_send_digest($realm);
```

```
    }
    // Zakończenie z błędem, jeśli nie można przeanalizować wartości skrótu
    $username = pc_parse_digest($_SERVER['PHP_AUTH_DIGEST'], $realm, $users);
    if ($username === flase) {
        pc_send_digest($realm);
    }
    // W skrócie została zawarta poprawna nazwa użytkownika
    return $username;
}

function pc_send_digest($realm) {
    header('http/1.O 4O1 Unauthorized');
    $nonce = md5(uniqid());
    $opaque = md5($realm);
    header("WWW-Authenticate: Digest realm=\"$realm\" qop=\"auth\" ".
           "nonce=\"$nonce\" opaque=\"$opaque\"");
    echo "Uzyskanie dostępu do strony wymaga podania poprawnej nazwy użytkownika
          i hasła.";
    exit;
}

function pc_parse_digest($digest, $realm, $users) {
    // W nagłówku skrótu muszą występować następujące parametry:
    // username, uri, qop, cnotce, nc oraz response
    $digest_info = array();
    foreach (array('username','uri','nonce','cnonce','response') as $part) {
        // Separatorem mogą być znaki (') lub (") lub brak jakiegokolwiek
        // znaku (w przypadku pól qop i nc)
        if (preg_match('/'.$part.'=([\'"]?)(.*?)\1/', $digest, $match)) {
            // Element został znaleziony i zostanie zapamiętany.
            $digest_info[$part] = $match[2];
        } else {
            // Jeśli element nie występuje, skrót jest niepoprawny.
            return false;
        }
    }
    // Sprawdzenie, czy została przekazana poprawna wartość qop
    if (preg_match('/qop=auth(,|$)/', $digest)) {
        $digest_info['qop'] = 'auth';
    } else {
        return false;
    }
    // Sprawdzenie, czy została przekazana poprawna liczba wartości nonce
    if (preg_match('/nc=([O-9a-f]{8})(,|$)/', $digest, $match)) {
        $digest_info['nc'] = $match[1];
    } else {
        return false;
    }

    // Wszystkie niezbędne wartości zostały wyodrębnione z nagłówka, można więc wykonać
    // obliczenia sprawdzające, czy dostarczone dane są poprawne.
    //
    // Wspomniane obliczenia są opisane w punktach 3.2.2, 3.2.2.1 oraz 3.2.2.2
    // dokumentu
    // RFC 2617.
    // Wykorzystywany algorytm do MD5
    $A1 = $digest_info['username'] . ':' . $realm . ':' .
          $users[$digest_info['username']];
    // Parametr qop ma wartość 'auth'
    $A2 = $_SERVER['REQUEST_METHOD'] . ':' . $digest_info['uri'];
    $request_digest = md5(implode(':', array(md5($A1), $digest_info['nonce'],
                      $digest_info['nc'], $digest_info['cnonce'], $digest_info['qop'],
                      md5($A2))));
```

```
    // Porównanie wartości przesłanej i obliczonej
    if ($request_digest != $digest_info['response']) {
        return false;
    }

    // Weryfikacja zakończona pomyślnie. Zwrócenie nazwy użytkownika.
    return $digest_info['username'];
}
?>
```

Osoby, które nie korzystają z interpretera PHP w wersji 5.1.0 lub późniejszej, ale używają interpretera PHP jako modułu serwera Apache, mogą korzystać z uwierzytelniania typu `Digest` dzięki takim rozwiązaniom, jak klasa `HTTPDigest` Paula Jamesa, dostępna pod adresem *http://www.peej.co.uk/projects/phphttpdigest.html*.

Ani mechanizmy prostego uwierzytelniania HTTP, ani mechanizmy uwierzytelniania typu `Digest` nie mogą być stosowane w przypadku, gdy program PHP funkcjonuje jako skrypt CGI. Jeżeli nie ma możliwości uruchomienia PHP jako modułu serwera, weryfikację użytkowników można oprzeć na danych cookies. Sposób ten omówiliśmy w recepturze 8.10.

Z uwierzytelnianiem HTTP wiąże się jeszcze jeden problem, jest nim brak — innej niż zakończenie pracy przeglądarki — możliwości wylogowania się. W podręczniku PHP zawarto kilka sugestii odnośnie sposobów wylogowywania, które niestety na różnych serwerach i w różnych przeglądarkach odnoszą różny skutek. Wspomniane informacje są dostępne pod adresem *http://www.php.net/features.http-auth*.

Istnieje jednak prosty sposób wymuszenia na użytkowniku ponownego logowania się po upływie określonego czasu. Rozwiązanie sprowadza się do umieszczenia wartości czasu w ciągu tekstowym określającym obszar witryny objęty uwierzytelnianiem. Jeżeli przeglądarka przemieszcza się po stronach w ramach danego obszaru witryny, to za każdym razem, kiedy musi przesłać nazwę użytkownika i hasło, dostarcza serwerowi tych samych, wcześniej wprowadzonych przez użytkownika danych. Zmiana nazwy obszaru objętego uwierzytelnianiem spowoduje, że przeglądarka będzie zmuszona do pobrania od użytkownika nowych parametrów uwierzytelniania. W listingu 8.21 został przedstawiony przykład mechanizmu umożliwiającego wylogowanie użytkowników w każdej dobie o północy.

Listing 8.21. Wymuszone wylogowanie w prostym uwierzytelnianiu HTTP

```
<?php
if (! pc_validate($_SERVER['PHP_AUTH_USER'],$_SERVER['PHP_AUTH_PW'])) {
    $realm = 'Moja strona do '.date('Y-m-d');
    header('WWW-Authenticate: Basic realm="'.$realm.'"');
    header('HTTP/1.0 401 Unauthorized');
    echo "Uzyskanie dostępu do strony wymaga podania poprawnej nazwy
          użytkownika i hasła.";
    exit;
}
?>
```

Poza tym istnieje możliwość ustalania dla każdego z użytkowników maksymalnego dopuszczalnego czasu przeglądania danej witryny. Wyeliminowana została tu konieczność zmiany nazwy obszaru witryny objętej uwierzytelnianiem. Pomysł polega na zapisaniu czasu, w którym użytkownik się zalogował lub pobrał chronioną stronę. Zaprezentowana w listingu 8.22 funkcja `pc_validate2()` zapisuje w bazie danych czas logowania i automatycznie przeprowadza wylogowanie klienta po piętnastu minutach od ostatniego pobrania chronionej strony.

Listing 8.22. Funkcja pc_validate2()

```
<?php
function pc_validate2($user,$pass) {
    $safe_user = strtr(addslashes($user),array('_' => '\_', '%' => '\%'));
    $r = mysql_query("SELECT password,last_access
                      FROM users WHERE user LIKE '$safe_user'");

    if (mysql_numrows($r) == 1) {
        $ob = mysql_fetch_object($r);
        if ($ob->password == $pass) {
            $now = time();
            if (($now - $ob->last_access) > (15 * 60)) {
                return false;
            } else {
                // zmień czas ostatniego pobrania strony
                mysql_query("UPDATE users SET last_access = NOW()
                             WHERE user LIKE '$safe_user'");
               return true;
            }
        }
    } else {
        return false;
    }
}
?>
```

Zobacz również

Podobne zagadnienia zostały omówione w recepturze 8.10. Sekcja internetowego podręcznika PHP poświęcona uwierzytelnianiu HTTP znajduje się pod adresem *http://www.php.net/features.http-auth*.

8.10. Uwierzytelnianie z wykorzystaniem danych cookie

Problem

Chcemy, aby procedura logowania była w większym stopniu uzależniona od kodu skryptu, aby można było wyświetlić niestandardowy formularz logowania.

Rozwiązanie

Sposobem na rozwiązanie problemu może być zapisywanie statusu uwierzytelnienia w danych cookie lub w postaci elementu sesji. Po prawidłowym zalogowaniu się użytkownika jego nazwa mogłaby być zapisana jako dane cookie. Procedura wymaga zastosowania szyfrowania nazwy użytkownika i dodatkowego utajnionego słowa, dzięki czemu klient nie będzie mógł spreparować cookie z informacjami uwierzytelniającymi i zawartą w nich nazwą użytkownika. Odpowiedni kod został przedstawiony w listingu 8.23.

Listing 8.23. Uwierzytelnianie z wykorzystaniem danych cookie

```
<?php
$secret_word = 'jedz szpinak';
if (pc_validate($_REQUEST['username'],$_REQUEST['password'])) {
```

```
    setcookie('login',
              $_REQUEST['username'].','.md5($_REQUEST['username'].$secret_word));
}
?>
```

Analiza

Dzięki zastosowaniu uwierzytelniania z wykorzystaniem danych cookie możliwe staje się przygotowanie własnego formularza logowania. Kod przykładowego formularza zawiera listing 8.24.

Listing 8.24. Prosty formularz logowania — uwierzytelnianie z wykorzystaniem danych cookie

```
<form method="post" action="login.php">
Nazwa użytkownika: <input type="text" name="username"> <br>
Hasło: <input type="password" name="password"> <br>
<input type="submit" value="Log In">
</form>
```

Do weryfikacji nazwy użytkownika i hasła można się posłużyć funkcją `pc_validate2()` przedstawioną w recepturze 8.18. Jedyna zmiana polega na tym, że parametry uwierzytelniania musimy przekazać za pomocą `$_REQUEST['username']` i `$_REQUEST['password']`, a nie jak poprzednio z użyciem `$_SERVER['PHP_AUTH_USER']` i `$_SERVER ['PHP_AUTH_PW']`. Jeśli weryfikacja hasła zakończy się pomyślnie, serwer powinien odesłać dane cookie, w których zostanie zawarta nazwa użytkownika oraz jego skrót wraz z utajnionym słowem. Zastosowanie funkcji skrótu (szyfrowanie) chroni witrynę przed ominięciem procedury logowania w wyniku przesłania danych cookie z nazwą użytkownika.

Po zalogowaniu, na każdej kolejnej stronie dokonywane będzie sprawdzenie, czy klient przesłał poprawne dane uwierzytelniające. Wynik operacji zadecyduje o tym, czy zalogowany wcześniej użytkownik będzie miał możliwość skorzystania ze strony. Kod opisanego rozwiązania znajduje się w listingu 8.25.

Listing 8.25. Weryfikacja danych uwierzytelniających zapisanych w pliku cookie

```
<?php
unset($username);
if ($_COOKIE['login']) {
    list($c_username,$cookie_hash) = split(',',$_COOKIE['login']);
    if (md5($c_username.$secret_word) == $cookie_hash) {
        $username = $c_username;
    } else {
        print "Przesłane dane cookie są niepoprawne.";
    }
}

if ($username) {
    print "Witaj, $username.";
} else {
    print "Witamy użytkownika anonimowego.";
}
?>
```

Wykorzystanie mechanizmów sesji pozwala na wyeliminowanie konieczności przesyłania danych cookie oraz zapisanie nazwy użytkownika i skrótu w plikach sesji. Koncepcja ta zakłada, że w chwili, gdy ktokolwiek się zaloguje, informacje o nim — zamiast w danych cookie — zostaną zapisane w nowo powołanej zmiennej sesji, tak jak to zostało pokazane w listingu 8.26.

Listing 8.26. Przechowywanie danych uwierzytelniających w zmiennych sesji

```
<?php
if (pc_validate($_REQUEST['username'],$_REQUEST['password'])) {
    $_SESSION['login'] =
        $_REQUEST['username'].','.md5($_REQUEST['username'].$secret_word));
}
?>
```

Kod odpowiedzialny za weryfikację danych (przedstawiony w listingu 8.27) niemal się nie zmienia. Zamiast tablicy `$_COOKIE` zastosowaliśmy w nim tablicę `$_SESSION`.

Listing 8.27. Weryfikacja danych uwierzytelniających

```
<?php
unset($username);
if ($_SESSION['login']) {
    list($c_username,$cookie_hash) = explode(',',$_SESSION['login']);
    if (md5($c_username.$secret_word) == $cookie_hash) {
        $username = $c_username;
    } else {
        print "Dane sesji zostały zmienione.";
    }
}
?>
```

Zastąpienie prostego uwierzytelniania HTTP uwierzytelnianiem z wykorzystaniem sesji lub danych cookies ułatwia realizację procedury wylogowania użytkowników. Cała operacja sprowadza się do usunięcia cookie z informacją o logowaniu lub usunięcia zmiennej sesji. Stosowanie zmiennych sesji do zapisywania informacji uwierzytelniających ma jeszcze jedną zaletę — pozwala na łączenie aktywności użytkowników po zalogowaniu z ich działalnością przed zalogowaniem i po wylogowaniu. Stosowanie prostego uwierzytelniania HTTP nie daje możliwości powiązania żądań zawierających nazwę użytkownika z żądaniami przesłanymi przez tego samego klienta przed podaniem nazwy. Porównanie adresów IP również się w tym przypadku nie sprawdza, gdyż użytkownik może pracować „ukryty" za firewallem lub serwerem proxy. Przy wykorzystaniu sesji możliwe jest takie zmodyfikowanie procedury logowania, by do dziennika pracy serwera zapisywane były również informacje o zależności pomiędzy identyfikatorem sesji a nazwą użytkownika. Kod realizujący wspomniane zadanie został przedstawiony w listingu 8.28.

Listing 8.28. Powiązanie danych o pracy użytkownika zalogowanego i niezalogowanego

```
<?php
if (pc_validate($_REQUEST['username'],$_REQUEST['password'])) {
    $_SESSION['login'] =
        $_REQUEST['username'].','.md5($_REQUEST['username'].$secret_word));
    error_log('Identyfikator sesji '.session_id().' zalogowany jako
              '.$_REQUEST['username']);
}
?>
```

Kod zaprezentowany w listingu 8.28 demonstruje sposób zapisywania komunikatów w dzienniku błędów. Nic nie stoi jednak na przeszkodzie, żeby taką samą procedurę wykorzystać do umieszczenia informacji w bazie danych, za pomocą której możliwe będzie późniejsze sporządzenie statystyk przeglądania strony i generowanego w ten sposób ruchu.

Zastosowanie do uwierzytelniania identyfikatora sesji ma również pewną wadę — sesje charakteryzują się niższym poziomem zabezpieczeń. Jeśli jeden z użytkowników będzie w stanie przewidzieć identyfikator sesji drugiego, może się zalogować na serwerze, podając się za drugiego

użytkownika. Aby utrudnić odgadnięcie identyfikatora, moduł sesji wyposażono w dwie dodatkowe, opcjonalne dyrektywy konfiguracyjne. Pierwsza z nich — `session.entropy_file` — definiuje ścieżkę do urządzenia lub pliku, który generuje wartości losowe (np. */dev/random* czy */dev/urandom*). Druga opcja — `session.entropy_length` — pozwala na określenie liczby bajtów, które odczytane z pliku wartości losowych posłużą do utworzenia identyfikatora sesji.

Jednak zabezpieczenie identyfikatora przed sfałszowaniem nie odniesie pożądanego skutku, jeśli zostanie on przechwycony w trakcie przesyłania otwartym tekstem pomiędzy serwerem a przeglądarką. Problem ten występuje również w przypadku prostego uwierzytelniania HTTP. Rozwiązaniem może być zastosowanie protokołu SSL, który zapobiega monitorowaniu danych przesyłanych w sieci (zagadnienie to zostało omówione w recepturze 18.13).

Zobacz również

Zagadnienia związane z rejestracją błędów zostały omówione w recepturach 8.9 i 20.9. Receptura 18.9 zawiera wskazówki odnośnie weryfikacji szyfrowanych danych. Dokumentacja na temat funkcji `setcookie()` znajduje się pod adresem *http://www.php.net/setcookie*, a na temat funkcji `md5()` — pod adresem *http://www.php.net/md5*.

8.11. Wymuszenie przesłania danych do przeglądarki

Problem

Chcemy wymusić przesłanie danych do przeglądarki. Operacja taka znajduje zastosowanie między innymi przy odwoływaniu się do powolnej bazy danych, kiedy użytkownik musi zostać poinformowany o postępach w realizacji jego żądania.

Rozwiązanie

Rozwiązanie problemu polega na zastosowaniu funkcji `flush()`, zaprezentowanej w listingu 8.29.

Listing 8.29. Przesłanie danych wyjściowych do przeglądarki

```
<?php
print 'Wyszukiwanie jednakowych płatków śniegu...';
flush();
$sth = $dbh->query(
    'SELECT rozmiar,COUNT(*) AS c FROM platki_sniegu GROUP BY rozmiar HAVING c > 1');
?>
```

Analiza

Wywołanie funkcji `flush()` powoduje przesłanie całej wygenerowanej przez PHP treści, która została do tej chwili zgromadzona w buforze przeznaczonym dla serwera WWW. Należy się jednak liczyć z możliwością wystąpienia opóźnienia w dotarciu informacji do przeglądarki, gdyż serwer WWW może posiadać własny mechanizm buforowania danych. Poza tym niektóre przeglądarki nie wyświetlają treści strony natychmiast po jej odebraniu, a pewne wersje Internet

Explorera nie prezentują strony, dopóki nie zgromadzą przynajmniej 256 bajtów danych. Aby wymusić na Internet Explorerze wyświetlenie informacji, należy umieścić na początku strony odpowiednią liczbę spacji, zgodnie z kodem zawartym w listingu 8.30.

Listing 8.30. Wymuszenie na przeglądarce Internet Explorer natychmiastowego wyświetlenia treści

```
<?php
print str_repeat(' ', 300);
print 'Wyszukiwanie jednakowych płatków śniegu...';
flush();
$sth = $dbh->query(
    'SELECT rozmiar,COUNT(*) AS c FROM platki_sniegu GROUP BY rozmiar HAVING c > 1');
?>
```

Zobacz również

Warto także zapoznać się z recepturą 23.13 oraz dokumentacją funkcji `flush()` dostępną pod adresem *http://www.php.net/flush*.

8.12. Buforowanie danych wyjściowych

Problem

Chcemy, aby rozpoczęcie generowania treści strony nastąpiło przed zakończeniem przesyłania nagłówków lub danych cookies.

Rozwiązanie

Na początku strony wywołamy funkcję `ob_start()`, a na końcu — `ob_end_flush()`. Pomiędzy wspomnianymi odwołaniami można umieszczać w dowolnej kolejności treść strony oraz polecenia przesyłania nagłówków. Dane te nie zostaną przesłane do momentu wywołania funkcji `ob_end_flush()`. Działanie mechanizmu można sprawdzić na podstawie kodu z listingu 8.31.

Listing 8.31. Buforowanie danych wyjściowych

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

Zastanawiam się jeszcze nad przesłaniem danych cookies.

<?php setcookie('czapla','niebieski'); ?>

Tak, decyzja o przesłaniu cookies była słuszna.

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

Analiza

Do przetwarzania danych wyjściowych można wykorzystać funkcję zwrotną, której nazwę przekazujemy jako argument wywołania funkcji `ob_start()`. Takie postępowanie jest wskazane, jeżeli zachodzi konieczność przeprowadzenia jakichkolwiek czynności końcowych, takich

jak ukrycie adresów e-mail przed wyszukującymi je robotami. Przykład tego rodzaju funkcji zwrotnej został przedstawiony w listingu 8.32.

Listing 8.32. Wykorzystanie funkcji zwrotnej w połączeniu z instrukcją ob_start()

```
<?php
function mangle_email($s) {
    return preg_replace('/([^@\s]+)@([-a-zO-9]+\.)+[a-z]{2,}/is',
                        '<$1@...>',
                        $s);
}

ob_start('mangle_email');
?>

Zabezpieczenie przed przesyłaniem niechcianych listów na adres zenon@przyklad.com!

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

Funkcja `mangle_email()` przekształca treść strony do postaci:

```
Zabezpieczenie przed przesyłaniem niechcianych listów na adres <zenon@...>!
```

Włączenie mechanizmu buforowania dla wszystkich generowanych stron wymaga przypisania dyrektywie konfiguracyjnej `output_buffering` wartości `On`:

```
output_buffering = On
```

Podobnie, możliwe jest określenie funkcji zwrotnej, która odpowiada za przetwarzanie informacji buforowanych przez każdą ze stron. Służy do tego dyrektywa `output_ handler`:

```
output_handler = mangle_email
```

Zdefiniowanie dyrektywy `output_handler` powoduje automatyczne włączenie `output_buffering`.

Zobacz również

Dokumentacja funkcji `ob_start()` znajduje się pod adresem *http://www.php.net/ob-start*, a funkcji `ob_end_flush()` — pod adresem *http:// www.php.net/ob-end-flush*. Informacje na temat buforowania danych wyjściowych zamieszczono również na stronie internetowej o adresie *http://www.php.net/outcontrol*.

8.13. Przesyłanie danych z użyciem kompresji gzip

Problem

Chcemy, aby dane przesyłane do przeglądarek wyposażonych w funkcję automatycznej dekompresji były kompresowane.

Rozwiązanie

Rozwiązanie problemu sprowadza się do uzupełnienia pliku *php.ini* o poniższą linię tekstu:

```
zlib.output_compression=1
```

Analiza

Informacja o tym, że przeglądarka akceptuje skompresowane strony, zamieszczana jest w przesyłanym do serwera nagłówku `Accept-Encoding`. Jeśli przeglądarka prześle nagłówek `Accept_Encoding: gzip` lub `Accept-Encoding: deflate`, a PHP został skompilowany z rozszerzeniem *zlib*, to dane wyjściowe, przed przekazaniem ich do przeglądarki, zostaną skompresowane za pomocą algorytmu określonego dyrektywą konfiguracyjną `zblib.output_compression`. Przeglądarka przed wyświetleniem strony dokona dekompresji danych.

Stopień kompresji można ustawiać za pomocą dyrektywy `zlib.output_compression_level`:

```
;najniższy poziom kompresji
zlib.output_compression_level=1

;najwyższy poziom kompresji
zlib.output_compression_level=9
```

Stosowanie wyższych poziomów kompresji powoduje zmniejszenie ilości przesyłanych danych przy jednoczesnym zwiększeniu obciążenia procesora serwera.

Zobacz również

Dokumentacja rozszerzenia *zlib* znajduje się pod adresem *http://www.php.net/zlib*.

8.14. Odczyt zmiennych środowiskowych

Problem

Chcemy pozyskać wartości zmiennych środowiskowych.

Rozwiązanie

Wspomniane wartości możemy pobrać z automatycznie tworzonej globalnej tablicy `$_ENV`, tak jak to zostało pokazane w listingu 8.33.

Listing 8.33. Odczyt zmiennej środowiskowej

```
<?php
$nazwa = $_ENV['USER'];
?>
```

Analiza

Zmienne środowiskowe są wartościami o określonych nazwach, które pozostają w związku z danym procesem. Na przykład, uzyskanie informacji o ścieżce dostępu do katalogu domowego użytkownika systemu Unix sprowadza się do odczytania wartości `$_ENV['HOME']` (zgodnie z instrukcjami zawartymi w listingu 8.34).

Listing 8.34. Odczyt innej zmiennej środowiskowej

```
<?php
print $_ENV['HOME'];  //katalog domowy użytkownika
?>
```

We wcześniejszych wersjach PHP operacja powoływania zmiennych odpowiadających wszystkim zmiennym środowiskowym była przeprowadzana automatycznie. Jednak od wersji 4.1.0, z uwagi na wydajność, w zalecanym pliku konfiguracyjnym *php.ini-recomended* opcja ta jest wyłączona. Z drugiej strony, w pliku *php.ini-dist* automatyczne pobieranie zmiennych środowiskowych pozostaje włączone, co ma zapewnić zgodność aplikacji z jej poprzednimi wersjami.

Tablica `$_ENV` jest tworzona jedynie w przypadku, gdy wartość dyrektywy konfiguracyjnej `variables_order` zawiera literę `E`. Jeśli tablica `$_ENV` nie jest dostępna, zmienne środowiskowe można odczytywać za pomocą funkcji `getenv()`, tak jak to zostało pokazane w listingu 8.35.

Listing 8.35. Wykorzystanie funkcji getenv()

```
<?php
$path = getenv('PATH');
?>
```

Funkcja `getenv()` jest niedostępna, jeżeli PHP działa jako moduł ISAPI.

Zobacz również

Zagadnienie przypisywania wartości zmiennym środowiskowym zostało omówione w podrozdziale 8.15. Dokumentacja funkcji `getenv()` jest dostępna pod adresem *http:// www.php.net/getenv*. Informacje o zmiennych środowiskowych można uzyskać na stronie internetowej *http://www.php.net/reserved.variables#reserved.variables.environment*.

8.15. Ustawianie wartości zmiennych środowiskowych

Problem

Chcemy umożliwić nadawanie wartości zmiennej środowiskowej z poziomu skryptu lub pliku konfiguracyjnego serwera. Definiowanie zmiennych środowiskowych w pliku konfiguracji serwera powinno pozwalać na ustalenie odrębnych parametrów konfiguracyjnych dla poszczególnych serwerów wirtualnych.

Rozwiązanie

Do ustalenia wartości zmiennej środowiskowej z poziomu skryptu służy funkcja `putenv()`, wykorzystana w kodzie listingu 8.36.

Listing 8.36. Ustawienie wartości zmiennej środowiskowej

```
<?php
putenv('ORACLE_SID=ORACLE');  //konfiguracja rozszerzenia oci
?>
```

Do ustalenia wartości zmiennej środowiskowej w pliku konfiguracyjnym serwera Apache — *httpd.conf* — służy dyrektywa `SetEnv`, której sposób użycia został przedstawiony w listingu 8.37.

Listing 8.37. Ustawienie zmiennej środowiskowej w pliku konfiguracyjnym serwera Apache

```
SetEnv DATABASE_PASSWORD hasło
```

Analiza

Przewagą metody zakładającej definiowanie zmiennych w pliku *httpd.conf* jest możliwość ustalenia bardziej restrykcyjnych praw odczytu pliku niż w przypadku praw do skryptu PHP. Pliki PHP muszą być odczytywane przez proces serwera WWW, przez co pozostali użytkownicy systemu zazwyczaj również mają do nich dostęp. Zapisanie haseł w pliku *httpd.conf* eliminuje konieczność umieszczania ich w plikach ogólnie dostępnych. Poza tym, jeżeli dany katalog macierzysty serwera jest dostępny pod kilkoma nazwami komputera, istnieje możliwość takiego skonfigurowania skryptu, by zasady jego funkcjonowania zmieniały się w zależności od zastosowanej nazwy komputera.

Załóżmy, że dysponujemy dwiema nazwami: *czlonkowie.przyklad.com* oraz *goscie.przyklad.com*. Korzystanie z pierwszego adresu wymaga uwierzytelnienia, ale oferuje dodatkowe możliwości. Użycie drugiej nazwy wiąże się z ograniczeniem swobody działania, lecz nie wymaga uwierzytelnienia. Rozwiązanie polegałoby wówczas na zastosowaniu kodu z listingu 8.38.

Listing 8.38. Zmiana sposobu działania aplikacji zależnie od wartości zmiennej środowiskowej

```
<?php
$version = $_ENV['SITE_VERSION'];

// odesłanie do strony http://goscie.przyklad.com w przypadku podania
// niepoprawnych informacji uwierzytelniających
if ('czlonkowie' == $version) {
    if (!authenticate_user($_REQUEST['username'], $_REQUEST['password'])) {
        header('Location: http://goscie.przyklad.com/');
        exit;
    }
}

include_once "${version}_header"; // załadowanie odpowiedniego nagłówka
?>
```

Zobacz również

Pobieranie wartości zmiennych środowiskowych zostało omówione w recepturze 8.14. Dokumentacja funkcji `getenv()` znajduje się pod adresem *http://www.php.net/putenv*. Informacje o zasadach ustalania wartości zmiennych środowiskowych serwera Apache dostępne są na stronie internetowej *http://httpd.apache.org/docs/mod/mod_env.html*.

8.16. Komunikacja w ramach serwera Apache

Problem

Chcemy ustanowić komunikację pomiędzy skryptem PHP a innymi elementami procesu obsługi żądania funkcjonującymi w ramach serwera Apache. Na przykład załóżmy, że potrzebujemy zmienić wartość w pliku *access_log*.

Rozwiązanie

Zastosujemy funkcję `apache_note()`, zgodnie z przykładem zawartym w listingu 8.39.

Listing 8.39. Komunikacja w ramach serwera Apache

```
<?php
// pobranie wartości
$session = apache_note('session');

// nadanie wartości
apache_note('session', $session);
?>
```

Analiza

Proces przetwarzania żądania klienta w Apache składa się z kilku etapów. PHP stanowi tylko jedno z ogniw całego łańcucha. Czynności serwera obejmują odwzorowanie adresu URL, uwierzytelnienie użytkownika, zapisanie żądania w dzienniku itd. Podczas analizowania żądania każda z procedur jego obsługi dysponuje zbiorem par typu klucz-wartość, zwanym **tablicą informacyjną** (ang. *notes table*). Funkcja `apache_note()` zapewnia dostęp do tablicy informacyjnej, umożliwiając pobieranie danych wprowadzonych tam przez procedury obsługi żądania, które zostały wykonane wcześniej i które zostawiły informacje przeznaczone dla procedur realizowanych w późniejszych etapach.

Przykładowo, zastosowanie modułu sesji do śledzenia poczynań użytkowników oraz zachowanie wartości zmiennych pomiędzy kolejnymi żądaniami pozwala na uzupełnienie procedury o analizę pliku dziennika, a dzięki temu na określenie liczby odwiedzin danej strony przypadającej na jednego użytkownika. Wykorzystanie funkcji `apache_note()` w połączeniu z modułem zapisu do dziennika umożliwia zapisywanie identyfikatora sesji każdego żądania bezpośrednio w pliku *access_log*. W takim przypadku w pierwszej kolejności trzeba dodać identyfikator sesji do tablicy informacyjnej, za co odpowiada kod przedstawiony w listingu 8.40.

Listing 8.40. Dodanie identyfikatora sesji do tablicy informacyjnej

```
<?php
// pobranie identyfikatora sesji i umieszczenie go w tablicy informacyjnej Apache
apache_note('session_id', session_id());
?>
```

Dalej konieczne jest zmodyfikowanie dyrektywy `LogFormat` pliku *httpd.conf* przez dodanie ciągu `%{session_id}n`. Znak `n` stanowi informację dla Apache, żeby wykorzystać zmienną zapisaną w tablicy informacyjnej przez inny moduł.

Jeżeli podczas kompilacji PHP dołączono opcję `--enable-memory-limit`, wartości szczytowe wykorzystania pamięci podczas realizacji każdego żądania zostaną zapisane w tablicy jako `mod_php_memory_usage`. Parametr ten można również zawrzeć w dyrektywie formatującej wpisy dziennika (`LogFormat`). Wystarczy dodać ciąg `%{mod_php_memory_usage}n`.

Zobacz również

Dokumentacja funkcji `apache_note()` znajduje się pod adresem *http://www.php.net/apache-note*. Informacje na temat funkcjonowania dzienników pracy serwera są dostępne na stronie internetowej *http://httpd.apache.org/docs/mod/mod_log_config.html*.

8.17. Program — aktywowanie i dezaktywowanie stron internetowych użytkowników

Każdy użytkownik, który występuje z prośbą o przydzielenie witryny internetowej, jest zobligowany do podania adresu poczty elektronicznej. Warto zatem sprawdzić, czy podany adres jest poprawny. Weryfikacja mogłaby polegać na odesłaniu listu na adres, który został podany w formularzu rejestracyjnym. Jeśli odbiorca listu nie otworzy w określonym czasie strony o specjalnym adresie URL, który został zamieszczony w treści listu, jego konto zostanie dezaktywowane.

Proponowany system składa się z trzech elementów. Pierwszy z nich stanowi program *notify-user.php*, którego działanie polega na przygotowywaniu i wysyłaniu listów do nowych użytkowników, w których będą oni proszeni o odwiedzenie strony o podanym adresie URL. Program jest przedstawiony w listingu 8.42. Drugą częścią systemu jest zamieszczona w listingu 8.43 strona *verify-user.php*, której zadaniem jest obsługa generowanych adresów URL i oznaczanie użytkowników jako poprawnie zweryfikowanych. Trzeci element stanowi program *delete-user.php*, dezaktywujący konta użytkowników, którzy nie wykorzystali przesłanego im adresu URL w określonym terminie. Program ten przedstawiono w listingu 8.44.

W listingu 8.41 zamieszczone są instrukcje SQL niezbędne do utworzenia tabeli, w której zostaną zapisane informacje o użytkownikach:

Listing 8.41. Kod SQL tabeli weryfikacji użytkowników

```
CREATE TABLE users (
 email VARCHAR(255) NOT NULL,
 created_on DATETIME NOT NULL,
 verify_string VARCHAR(16) NOT NULL,
 verified TINYINT UNSIGNED
);
```

Większość programistów zapewne zechce zgromadzić trochę więcej informacji na temat każdego z użytkowników, niemniej jednak pola przedstawione w listingu 8.41 są wystarczające do przeprowadzenia weryfikacji. Podczas tworzenia konta użytkownika należy zapisać jego dane w tabeli `users` oraz przesłać do niego list z informacjami o sposobie uaktywnienia konta. Przy pisaniu kodu listingu 8.42 przyjęliśmy założenie, że adres poczty elektronicznej użytkownika jest przechowywany w zmiennej `$email`.

Listing 8.42. Program notify-user.php

```
<?php
// Połączenie z bazą danych
$db = new PDO('sqlite:users.db');

$email = 'david';

// Generowanie ciągu uwierzytelniającego — verify_string
$verify_string = '';
for ($i = 0; $i < 16; $i++) {
    $verify_string .= chr(mt_rand(32,126));
}

// Wprowadzenie danych użytkownika do bazy danych
// Wykorzystanie funkcji datetime() charakterystycznej dla rozszerzenia SQLite
$sth = $db->prepare("INSERT INTO users ".
                    "(email, created_on, verify_string, verified) ".
                    "VALUES (?, datetime('now'), ?, 0)");
$sth->execute(array($email, $verify_string));

$verify_string = urlencode($verify_string);
$safe_email = urlencode($email);

$verify_url = "http://www.przyklad.com/verify.php";

$mail_body=<<<_MAIL_
To $email:

Proszę kliknąć poniższy odsyłacz w celu potwierdzenia chęci założenia konta:

$verify_url?email=$safe_email&verify_string=$verify_string

Niepotwierdzenie chęci założenia konta w ciągu siedmiu dni spowoduje jego usunięcie.
_MAIL_;

// mail($email, "Prośba o potwierdzenie chęci założenia konta", $mail_body);
print "$email, $mail_body";
```

Strona weryfikująca użytkowników — do której użytkownicy są kierowani po kliknięciu odsyłacza zawartego w treści listu — uaktualnia tabelę `users`. Warunkiem jest dostarczenie poprawnych informacji. Kod programu znajduje się w listingu 8.43.

Listing 8.43. Program verify-user.php

```
<?php
// Połączenie z bazą danych
$db = new PDO('sqlite:users.db');
$sth = $db->prepare('UPDATE users SET verified = 1 WHERE email = ? '.
                    'AND verify_string = ? AND verified = 0');
$res = $sth->execute(array($_GET['email'], $_GET['verify_string']))l
var_dump($res, $sth->rowCount());
if (! $res) {
          print "Twoje konto zostało pomyślnie zweryfikowane. Dziękujemy.";
} else {
    print "Niestety, prośba o aktywację konta została odrzucona.";
}
?>
```

Status użytkownika podlega uaktualnieniu jedynie w przypadku, gdy adres poczty elektronicznej i ciąg uwierzytelniający zgadzają się z wartościami zapisanymi w bazie danych i tylko jeśli nie zostały wcześniej pomyślnie zweryfikowane. Ostatnim elementem systemu jest krótki

program usuwający użytkowników, którzy w określonym czasie nie przeprowadzili procedury weryfikacji. Jego kod jest przedstawiony w listingu 8.44.

Listing 8.44. Program delete-user.php

```
<?php
// Połączenie z bazą danych
$db = new PDO('sqlite:users.db');

$window = '-7 days';

$sth = $db->prepare("DELETE FROM users WHERE verified = 0 AND ".
                    "created_on < datetime('now',?)");
$res = $sth->execute(array($window));

if ($res) {
    print "Usunięto konta $deleted_users użytkowników.\n";
} else {
    print "Nie można usunąć informacji o użytkownikach.";
}
```

Program przedstawiony w listingu 8.44 należy uruchamiać raz dziennie w celu przeanalizowania tabeli `users` i odrzucenia użytkowników, którzy nie zostali pomyślnie zweryfikowani. Zmieniając wartość zmiennej `$window`, można regulować czas, w jakim użytkownicy powinni poddać się procedurze. Trzeba jednak pamiętać o uwzględnieniu nowej wartości w komunikacie przesyłanym pocztą elektroniczną.

8.18. Prosty serwis Wiki

Przedstawiony w listingu 8.45 program łączy w sobie różne techniki programowania opisane w tym rozdziale i udostępnia w pełni funkcjonalny system Wiki — serwis, którego strony mogą być edytowane przez użytkowników. Została tu zachowana struktura kodu charakterystyczna dla nieskomplikowanych skryptów PHP. W pierwszej części programu definiowane są różne parametry konfiguracyjne. W dalszej części występuje sekcja `if-else`, odpowiedzialna za ustalenie celu wywołania skryptu (wyświetlenie strony, zapisanie wprowadzonych zmian itd.) na podstawie wartości zapisanych w ciągu URL. Pozostała część kodu składa się z funkcji wywoływanych z poziomu wspomnianej sekcji `if-else`. Są to funkcje wyświetlenia nagłówka i stopki strony, pobrania treści strony oraz wyświetlenia formularza edycyjnego.

Program serwisu Wiki bazuje na zewnętrznej bibliotece PHP Markdown (utworzonej przez Michela Fortina), przeznaczonej do przekształcania poręcznej i zwartej składni Markdown w kod HTML. Biblioteka PHP Markdown jest dostępna pod adresem *http://www.michelf.com/projects/php-markdown/*.

Listing 8.45. Prosty serwis Wiki

```
<?php

// W programie została wykorzystana funkcja Markdown dostępna pod adresem
// http://www.michelf.com/projects/php-markdown/
// Umożliwia ona opisywanie treści w sposób charakterystyczny dla serwisów Wiki.
require_once 'markdown.php';

// Katalog, w którym przechowywane są strony Wiki.
// Serwer WWW musi mieć możliwość zapisywania w nim danych.
define('PAGEDIR',dirname(__FILE__) . '/strony');
```

```
// Pobranie nazwy strony lub wykorzystanie nazwy domyślnej
$page = isset($_GET['page']) ? $_GET['page'] : 'Strona główna';

// Ustalenie celu uruchomienia skryptu – wyświetlenie formularza edycyjnego, zapis
// formularza edycyjnego lub wyświetlenie strony

// Wyświetlenie żądanego formularza edycyjnego
if (isset($_GET['edit'])) {
    pageHeader($page);
    edit($page);
    pageFooter($page, false);
}
// Zapis przesłanego formularza edycyjnego
else if (isset($_POST['edit'])) {
    file_put_contents(pageToFile($_POST['page']), $_POST['contents']);
    // Skierowanie do standardowego widoku zmodyfikowanej strony
    header('Location: http://'.$_SERVER['HTTP_HOST'] . $_SERVER['SCRIPT_NAME'] .
           '?page='.urlencode($_POST['page']));
    exit();
}
// Wyświetlenie strony
else {
    pageHeader($page);
    // Jeśli strona istnieje, zostanie wyświetlona wraz z odsyłaczem "Edycja" w stopce
    if (is_readable(pageToFile($page))) {
        // Pobranie treści strony z pliku, w którym jest zapisana
        $text = file_get_contents(pageToFile($page));
        // Przekształcenie składni Markdown (za pomocą funkcji Markdown()
        // z biblioteki markdown.php)
        $text = Markdown($text);
        // Przekształcenie ciągów [odsyłacz] w odsyłacze do innych stron Wiki
        $text = wikiLinks($text);
        // Wyświetlenie strony
        echo $text;
        // Wyświetlenie stopki
        pageFooter($page, true);
    }
    // Jeśli strona nie istnieje, zostanie wyświetlony fomularz edycyjny
    // oraz stopka bez odsyłacza "Edycja"
    else {
        edit($page, true);
        pageFooter($page, false);
    }
}

// Nagłówek strony -- nieskomplikowany, zawierający jedynie tytuł oraz standardowe
// ozdobniki HTML
function pageheader($page) { ?>
<html>
<head>
<title>Wiki: <?php echo htmlentities($page) ?></title>
</head>
<body>
<h1><?php echo htmlentities($page) ?></h1>
<hr/>
<?php
}

// Stopka strony – data ostatniej modyfikacji, opcjonalny odsyłacz "Edycja" oraz
// odsyłacz do głównej strony serwisu
function pageFooter($page, $displayEditLink) {
    $timestamp = @filemtime(pageToFile($page));
    if ($timestamp) {
        $lastModified = strftime('%c', $timestamp);
```

```
    } else {
        $lastModified = 'Nigdy';
    }
    if ($displayEditLink) {
        $editLink = ' - <a href="?page='.urlencode($page).'&edit=true">Edycja</a>';
    } else {
        $editLink = '';
    }
?>
<hr/>
<em>Data ostatniej modyfikacji: <?php echo $lastModified ?></em>
<?php echo $editLink ?> - <a href="<?php echo $_SERVER['SCRIPT_NAME'] ?>">Strona
główna</a>
</body>
</html>
<?php
}

// Wyświetlenie formularza edycyjnego. Jeśli strona już istnieje, formularz zostanie
// uzupełniony dotychczasową treścią
function edit($page, $isNew = false) {
    if ($isNew) {
        $contents = '';
?>
<p><b>Strona nie została jeszcze utworzona.</b> Aby ją utworzyć, wypełnij pole treści
i kliknij przycisk <b>Zapisz</b>.</p>
    <?php } else {
        $contents = file_get_contents(pageToFile($page));
    }
?>
<form method='post' action='<?php echo htmlentities($_SERVER['SCRIPT_NAME']) ?>'>
<input type='hidden' name='edit' value='true'/>
<input type='hidden' name='page' value='<?php echo htmlentities($page) ?>'/>
<textarea name='contents' rows='20' cols='60'>
<?php echo htmlentities($contents) ?></textarea>
<br/>
<input type='submit' value='Zapisz'/>
</form>
<?php
}

// Przekształcenie przesłanej strony w plik. Zastosowanie funkcji md5() stanowi
// zabezpieczenie przed wystąpieniem niedozwolonych znaków w zmiennej $page.
function pageToFile($page) {
    return PAGEDIR.'/'.md5($page);
}

// Przekształcenie tekstu [coś] w odsyłacz HTML do strony Wiki "coś".
function wikiLinks($page) {
    if (preg_match_all('/\[([^\]]+?)\]/', $page, $matches, PREG_SET_ORDER)) {
        foreach ($matches as $match) {
            $page = str_replace($match[0], '<a href="'.$_SERVER['SCRIPT_NAME'].
'?page='.urlencode($match[1]).'">'.htmlentities($match[1]).'</a>', $page);
        }
    }
    return $page;
}
?>
```