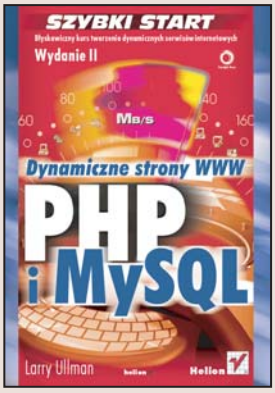
PHP i MySQL. Dynamiczne strony WWW. Szybki start. Wydanie II



Larry Ullman
Tłumaczenie: Jaromir Senczyk i Grzegorz Werner
na podstawie tłumaczenia Michała Dadana i Piotra Pilcha
ISBN: 83-246-0207-0
Tytuł oryginału: PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide (2nd Edition)
Format: B5, stron: 688

Błyskawiczny kurs tworzenia dynamicznych serwisów internetowych

Dynamiczne strony WWW spotykamy codziennie, korzystając z internetu. Portale, sklepy internetowe, gry sieciowe – wszystkie te witryny korzystają z baz danych i skryptów wykonywanych po stronie serwera. Technologii umożliwiających realizację tego typu witryn WWW jest kilka. Wśród nich zasłużoną popularnością cieszy się „duet" o ogromnych możliwościach – język skryptowy PHP i baza danych MySQL. Te dostępne nieodpłatnie narzędzia wykorzystywane są przez tysiące twórców witryn WWW. Dołącz do nich!

Książka „PHP i MySQL. Dynamiczne strony WWW. Szybki start. Wydanie II" to kolejna edycja doskonałego przewodnika po tajnikach tworzenia witryn internetowych za pomocą tych technologii. Znajdziesz w niej wszystkie informacje niezbędne do rozpoczęcia projektowania własnych dynamicznych stron WWW – od podstaw programowania i korzystania z baz danych, poprzez wykorzystywanie sesji i plików cookie, aż do zaawansowanych technik autoryzowania użytkowników i budowania aplikacji e-commerce. Każde zagadnienie jest przedstawione na praktycznym przykładzie, co doskonale pomoże Ci w przyswojeniu wiedzy.

- Podstawowe elementy skryptów PHP
- Obsługa formularzy HTML
- Tworzenie i stosowanie funkcji
- Projektowanie baz danych
- Operacje na danych
- Wykrywanie i usuwanie błędów w skryptach
- Łączenie skryptów PHP z bazą danych
- Stosowanie plików cookie i mechanizmów zarządzania sesjami
- Zabezpieczanie i szyfrowanie danych
- Zarządzanie treścią strony
- Autoryzowanie użytkowników
- Projektowanie sklepów internetowych

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

Spis treści

Tworzenie dynamicznych stron WWW

3

Teraz, gdy opanowałeś już podstawy PHP, czas rozpocząć tworzenie naprawdę dynamicznych stron WWW. W porównaniu ze stronami statycznymi, które dominowały na początku istnienia internetu, są one łatwiejsze do utrzymania i bardziej interaktywne, a ich wygląd może się zmieniać w zależności od sytuacji.

W tym rozdziale omawiam bardzo wiele rozwiązań, które wykorzystuje się do budowy dynamicznych aplikacji internetowych. Piszę między innymi o plikach zewnętrznych, obsłudze formularzy innymi sposobami, pisaniu i wykorzystywaniu własnych funkcji, wysyłaniu poczty elektronicznej i stosowaniu funkcji `date()`. Wszystkie wymienione elementy są wykorzystywane przez bardziej zaawansowane aplikacje internetowe.

Wykorzystywanie plików zewnętrznych

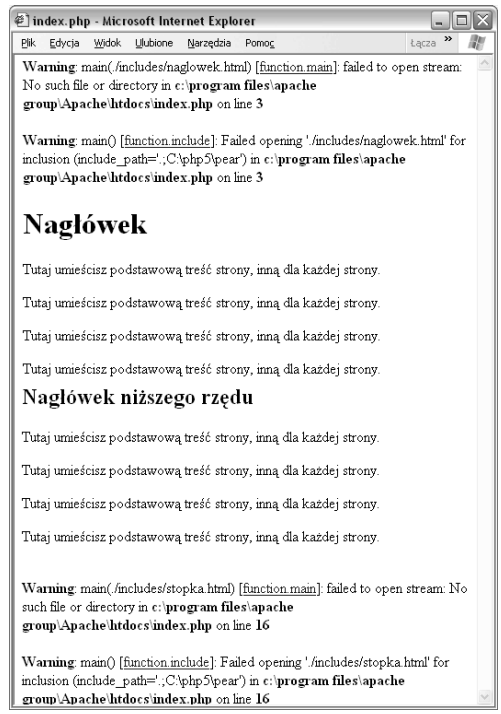
Wszystkie zaprezentowane dotychczas skrypty miały postać pojedynczych plików, w których zapisany był cały kod HTML i PHP. Gdy zaczniesz tworzyć bardziej rozbudowane witryny, zorientujesz się, że takie podejście ma wiele ograniczeń. Na szczęście PHP obsługuje pliki zewnętrzne, co pozwala Ci rozbić kod na kilka części. Dzięki temu będziesz mógł oddzielić kod HTML od kodu PHP i wyodrębnić ze skryptu tę jego część, która odpowiada za najczęściej wykonywane operacje.

PHP ma cztery funkcje obsługujące pliki zewnętrzne: `include()`, `include_once()`, `require()` i `require_once()`. Stosuje się je, umieszczając w skrypcie PHP instrukcje tego typu:

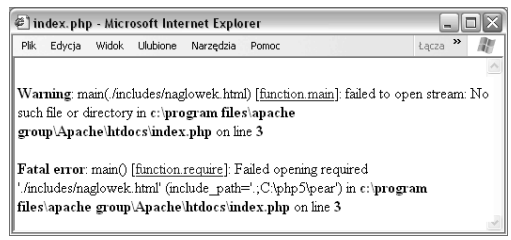
```
include_once("nazwapliku.php");
require('/ścieżka/do/pliku/nazwapliku.html');
```

Działają one w ten sposób, że biorą całą zawartość pliku o podanej nazwie i wstawiają ją do oryginalnego skryptu w miejscu swego wystąpienia. PHP zakłada, że kod występujący w plikach zewnętrznych jest kodem HTML i że powinien zostać przesłany bezpośrednio do przeglądarki (chyba że jest on otoczony znacznikami oznaczającymi kod PHP).

W poprzednich wersjach PHP funkcję `include()` stosowało się w nieco innych okolicznościach niż `require()`. Obecnie ich zastosowanie jest takie samo, choć różnią się one swym zachowaniem w sytuacjach awaryjnych. Jeżeli z jakichś względów funkcja `include()` nie będzie mogła wczytać pliku, w przeglądarce internetowej zostanie wyświetlony komunikat o błędzie (rysunek 3.1), ale skrypt będzie w dalszym ciągu wykonywany. Jeżeli natomiast załadowanie pliku nie uda się funkcji `require()`, po wyświetleniu informacji o błędzie skrypt zostanie zatrzymany (rysunek 3.2).



Rysunek 3.1. *Dwa wywołania funkcji include() zakończone niepowodzeniem*



Rysunek 3.2. *Przy pierwszym niepowodzeniu funkcji require() skrypt zostanie zatrzymany, a w przeglądarce pojawi się komunikat o błędzie*

Obie funkcje mają też odmianę `_once()`, która gwarantuje, że dany plik zewnętrzny zostanie dołączony do skryptu tylko jeden raz, nawet jeśli programista popełni błąd i umieści w skrypcie kilka poleceń dołączających.

```
require_once('nazwapliku.html');
```

Czas na pierwszy przykład. Oddzielę w nim kod HTML odpowiedzialny za formatowanie tekstu od kodu PHP, wykorzystując mechanizm dołączania plików. Dzięki temu wszystkie kolejne przykłady w tym rozdziale będą mogły zwracać wyniki w tej samej postaci, a ja nie będę musiał przepisywać za każdym razem tych samych fragmentów kodu. Powstanie w ten sposób system szablonów zapewniający spójność i łatwe zarządzanie rozbudowanymi aplikacjami. W przykładach skoncentruję się na kodzie PHP. Powinieneś również przeczytać informacje umieszczone w ramce „Struktura witryny", dzięki którym zrozumiesz schemat organizacji plików. Jeśli będziesz mieć pytania dotyczące *CSS* (*Cascading Style Sheets*) lub (X)HTML używanych w przykładach, to skorzystaj z odnośników podanych w dodatku C, „Zasoby internetowe".

Struktura witryny

Gdy zaczynasz wykorzystywać w swych aplikacjach internetowych pliki zewnętrzne, całościowa struktura witryny nabiera większego znaczenia. Projektując serwis internetowy, powinieneś brać pod uwagę trzy czynniki:

- łatwość utrzymywania,
- bezpieczeństwo,
- łatwość poruszania się po witrynie.

Wykorzystanie plików zewnętrznych do przechowywania standardowych procedur PHP, *CSS*, JavaScript i HTML bardzo upraszcza utrzymywanie witryny, ponieważ cały wspólny kod jest przechowywany tylko w jednym miejscu. W kolejnych przykładach będę często tworzył specjalne katalogi na pliki zewnętrzne i trzymał je odrębnie od zasadniczych skryptów.

W przypadku dokumentów niezawierających poufnych danych, takich jak szablony HTML, zalecam stosowanie rozszerzenia *.inc* lub *.html*, natomiast pliki wymagające większego bezpieczeństwa (przechowujące na przykład informacje potrzebne do połączenia się z bazą danych) powinny mieć rozszerzenia *.php.* Używaj nawet obu rozszerzeń (czyli *.inc* i *.html* lub *.php*), aby wskazać, że dany plik jest plikiem zewnętrznym określonego typu.

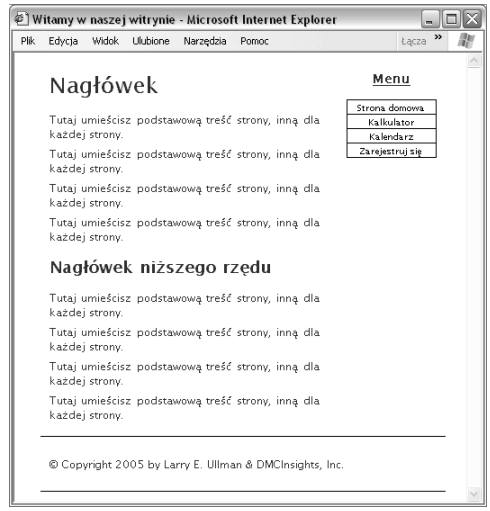
Powinieneś też projektować swoje witryny w taki sposób, aby użytkownicy mogli się po nich łatwo poruszać i to zarówno posługując się łączami, jak i ręcznie wpisywanymi adresami URL. Staraj się nie tworzyć zbyt wielu poziomów folderów i nie używać trudnych do zapamiętania nazw plików i katalogu. Nie mieszaj też wielkich liter z małymi i nie stosuj w nazwach znaków przestankowych.

Aby wykorzystać pliki zewnętrzne:

1. Stwórz w edytorze tekstów lub programie typu WYSIWYG projekt strony HTML (listing 3.1 i rysunek 3.3).

   Na początek zaprojektuję wygląd mojej aplikacji HTML (jest on całkowicie niezależny od kodu PHP). Za pomocą komentarzy oznaczę tę część projektu, która będzie inna dla każdej strony.

   Uwaga: aby zaoszczędzić miejsca, nie zamieściłem pliku CSS, który decyduje o wyglądzie strony. Możesz go ściągnąć razem z kodami przykładów z serwera ftp lub uruchamiać przykład bez tego pliku (szablon będzie działać, ale jego wygląd nie będzie zbyt estetyczny).



Rysunek 3.3. *Projekt strony HTML oglądany w przeglądarce internetowej (wykorzystujący HTML i CSS, ale jeszcze nie PHP)*

Listing 3.1. *Szablon stron WWW generowanych w tym rozdziale*

Listing

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <meta h1ttp-equiv="content-type" content="text/html; charset=iso-8859-2" />
6     <title>Witamy w naszej witrynie!</title>
7     <style type="text/css" media="all">@import "./includes/layout.css";</style>
8   </head>
9   <body>
10  <div id="wrapper"><!-- Zgodnie ze stylem CSS. -->
11
12    <div id="content"><!-- Zgodnie ze stylem CSS. -->
13
14      <div id="nav"><!-- Sekcja łączy -->
15        <h3>Menu</h3>
16        <ul>
17        <li class="navtop"><a href="index.php" title="Przejdź do strony domowej">Strona
          domowa</a></li>
18        <li><a href="kalkulator.php" title="Użyj kalkulatora">Kalkulator</a></li>
19        <li><a href="kalendarz.php" title="Wypróbuj działanie formularza wyboru
          daty">Kalendarz</a></li>
20        <li><a href="rejestracja.php" title="Zarejestruj się">Zarejestruj się</a></li>
21        </ul>
22      </div>
23
24      <!-- Początek treści specyficznej dla danej strony. -->
25      <h1 id="mainhead">Nagłówek</h1>
26      <p>Tutaj umieścisz podstawową treść strony, inną dla każdej strony.</p>
27      <p>Tutaj umieścisz podstawową treść strony, inną dla każdej strony.</p>
28      <p>Tutaj umieścisz podstawową treść strony, inną dla każdej strony.</p>
```

Listing 3.1. *Szablon stron WWW generowanych w tym rozdziale — ciąg dalszy*

Listing

```
29  <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
30  <h2>Nagłówek niższego rzędu</h2>
31  <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
32  <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
33  <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
34  <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
35  <!-- Koniec treści specyficznej dla
    danej strony. -->
36
37    </div><!-- Koniec DIV "treści". -->
38
39  <div id="footer"><p>&copy; Copyright
    2005 by Larry E. Ullman
    &amp;DMCInsights,
    Inc.</p></div>
40
41  </div><!-- Koniec DIV "obudowy". -->
42  </body>
43   </html>
```

Listing 3.2. *Początek każdej strony będzie przechowywany w pliku nagłówkowym*

Listing

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD
    XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/
       xhtml1-transitional.dtd">
3   <html
    xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="en" lang="en">
4   <head>
5     <meta http-equiv="content-type"
        content="text/html;
        charset=iso-8859-2" />
```

2. Skopiuj wszystko, począwszy od pierwszego wiersza aż do początku kodu specyficznego dla danej strony i wklej to do nowego dokumentu (listing 3.2).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type"
  content="text/html; charset=iso-8859-2" />
  <title><?php echo $page_title; ?></title>
  <style type="text/css" media="all">@import
  "./includes/layout.css";</style>
</head>
<body>
<div id="wrapper">
  <div id="content">
    <div id="nav">
      <h3>Menu</h3>
      <ul>
      <li class="navtop"><a href="index.php"
      title="Przejdź do strony domowej">Strona
      domowa</a></li>
      <li><a href="kalkulator.php" title="Użyj
      kalkulatora">Kalkulator</a></li>
      <li><a href="kalendarz.php"
      title="Wypróbuj działanie formularza
      wyboru daty">Kalendarz</a></li>
      <li><a href="rejestracja.php"
      title="Zarejestruj się">Zarejestruj
      się</a></li>
      </ul>
</div>
<!-- Skrypt 3.2 - naglowek.html -->
```

Pierwszy plik będzie zawierał początkowe znaczniki HTML (od `DOCTYPE`, przez `head`, aż do początku „ciała” strony).

3. Zmień wiersz zawierający tytuł strony na:

```
<title><?php echo $page_title; ?></title>
```

Chcę, aby tytuł strony (pojawiający się na górze w przeglądarce internetowej; patrz rysunek 3.3) był na każdej stronie inny. Dlatego też będę przypisywał go do zmiennej, której zawartość będzie wyświetlana przez PHP.

4. Zapisz plik pod nazwą *naglowek.html*.

Nazwy plików zewnętrznych mogą mieć całkowicie dowolne rozszerzenia. Niektórzy programiści preferują rozszerzenie *.inc*, ponieważ sugeruje ono, że mamy do czynienia z plikiem dołączanym (ang. *included file*). W tym przypadku mógłbyś również użyć rozszerzenia .inc.html, wskazującego, że dołączany plik zawiera kod HTML (a nie PHP).

Listing 3.2. *Początek każdej strony będzie przechowywany w pliku nagłówkowym — ciąg dalszy*

Listing

```
6     <title><?php echo $page_title;
        ?></title>
7     <style type="text/css" media="all">
      @import "./includes/layout.css";
      </style>
8   </head>
9   <body>
10  <div id="wrapper"><!-- Zgodnie ze
      stylem CSS. -->
11
12    <div id="content"><!-- Zgodnie ze
        stylem CSS. -->
13
14      <div id="nav"><!-- Sekcja łączy -->
15        <h3>Menu</h3>
16        <ul>
17        <li class="navtop"><a href=
          "index.php" title="Przejdź do
          strony domowej">
          Strona domowa</a></li>
18        <li><a href="kalkulator.php"
          title="Użyj kalkulatora">
          Kalkulator</a></li>
19        <li><a href="kalendarz.php"
          title="Wypróbuj działanie
          formularza wyboru daty">
          Kalendarz</a></li>
20        <li><a href="rejestracja.php"
          title="Zarejestruj się">
          Zarejestruj się</a></li>
21        </ul>
22      </div>
23      <!-- Skrypt 3.2 - naglowek.html -->
24      <!-- Początek treści specyficznej
        dla danej strony. -->
```

Listing 3.3. *Zakończenie każdej strony będzie przechowywane w pliku stopki*

Listing

```
1         <!-- Koniec treści specyficznej
          dla danej strony. -->
2       <!-- Skrypt 3.3 - stopka.html -->
3     </div><!-- Koniec DIV "treści". -->
4
5     <div id="footer"><p>&copy; Copyright
      2005 by Larry E. Ullman &amp;
      DMCInsights, Inc.</p></div>
6
7   </div><!-- Koniec DIV "obudowy". -->
8   </body>
9   </html>
```

Listing 3.4. *Ten skrypt tworzy stronę WWW, wykorzystując szablon przechowywany w plikach zewnętrznych*

Listing

```
1   <?php # Skrypt 3.4 - index.php
2   $page_title = 'Witamy w naszej
      witrynie!';
3   include ('./includes/naglowek.html');
4   ?>
5   <h1 id="mainhead">Nagłówek</h1>
6   <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
7   <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
8   <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
9   <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
10  <h2>Nagłówek niższego rzędu</h2>
11  <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
12  <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
13  <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
14  <p>Tutaj umieścisz podstawową treść
    strony, inną dla każdej strony.</p>
15  <?php
16  include ('./includes/stopka.html');
17  ?>
```

5. Przejdź do oryginalnego szablonu strony i skopiuj wszystko od końca części specyficznej dla konkretnej strony do końca pliku. Następnie wklej to do nowego pliku (listing 3.3).

```
<!-- Skrypt 3.3 - stopka.html -->
</div><!-- Koniec DIV "treści". -->
  <div id="footer"><p>&copy; Copyright
  2005 by Larry E. Ullman &amp;
  DMCInsights, Inc.</p></div>
</div><!-- Koniec DIV "obudowy". -->
</body>
</html>
```

Plik stopki zawiera ostatnie znaczniki formatujące i znaczniki zamykające dokument HTML.

6. Zapisz plik pod nazwą *stopka.html*.

7. Utwórz w edytorze tekstów nowy dokument PHP (listing 3.4).

```
<?php # Skrypt 3.4 - index.php
```

Ponieważ większość znaczników formatujących znajduje się w plikach zewnętrznych, możemy rozpocząć dokument od znaczników PHP, a nie HTML.

8. Nadaj zmiennej `$page_title` odpowiednią wartość i dołącz nagłówek HTML.

```
$page_title = 'Witaj!';
include ('./includes/naglowek.inc');
```

Dzięki zmiennej `$page_title` każda strona wygenerowana z wykorzystaniem tego szablonu może mieć inny tytuł. Ponieważ jej wartość jest określana przed dołączeniem pliku nagłówkowego, dostęp do niej ma ten ostatni. Pamiętaj, że efektem wykonania wiersza kodu zawierającego wywołanie funkcji `include()` jest umieszczenie zamiast niego całej zawartości dołączanego pliku.

9. Pozamykaj znaczniki PHP i skopiuj z szablonu stron kod specyficzny dla danej strony.

```
?>
<h1 id="mainhead">Nagłówek</h1>
<p>Tutaj umieścisz podstawową treść strony,
inną dla każdej strony.</p>
<p>Tutaj umieścisz podstawową treść strony,
inną dla każdej strony.</p>
<p>Tutaj umieścisz podstawową treść strony,
inną dla każdej strony.</p>
<p>Tutaj umieścisz podstawową treść strony,
inną dla każdej strony.</p>
<h2>Nagłówek niższego rzędu</h2>
<p>Tutaj umieścisz podstawową treść strony,
inną dla każdej strony.</p>
<p>Tutaj umieścisz podstawową treść strony,
inną dla każdej strony.</p>
<p>Tutaj umieścisz podstawową treść strony,
inną dla każdej strony.</p>
<p>Tutaj umieścisz podstawową treść strony,
inną dla każdej strony.</p>
```

Te informacje można też przesłać do przeglądarki za pośrednictwem funkcji `echo()`, ale ponieważ nie występują w nich żadne treści generowane dynamicznie, szybszym i wydajniejszym rozwiązaniem jest chwilowe opuszczenie znaczników PHP.

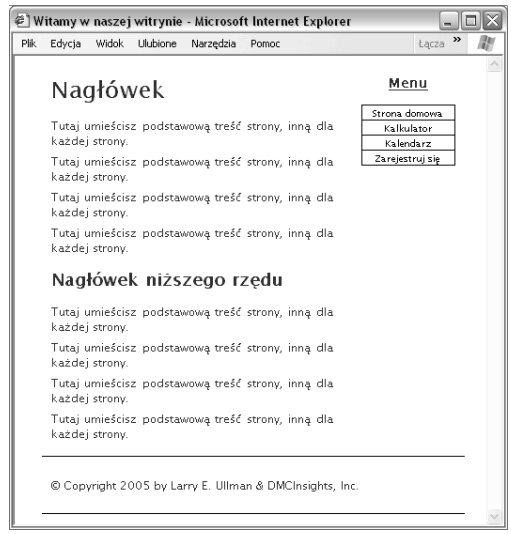
10. Stwórz w kodzie ostatnią sekcję PHP i dołącz plik stopki.

```
<?php
include ('./includes/stopka.inc');
?>
```

11. Zapisz dokument pod nazwą *index.php*, wgraj go na serwer razem z plikiem *stopka.html*.

12. Utwórz podkatalog *includes* w tym samym katalogu, w którym znajduje się *index.php*. Następnie umieść w katalogu includes pliki *naglowek.html*, *stopka.html* i *layout.css* (załadowany z serwera ftp).

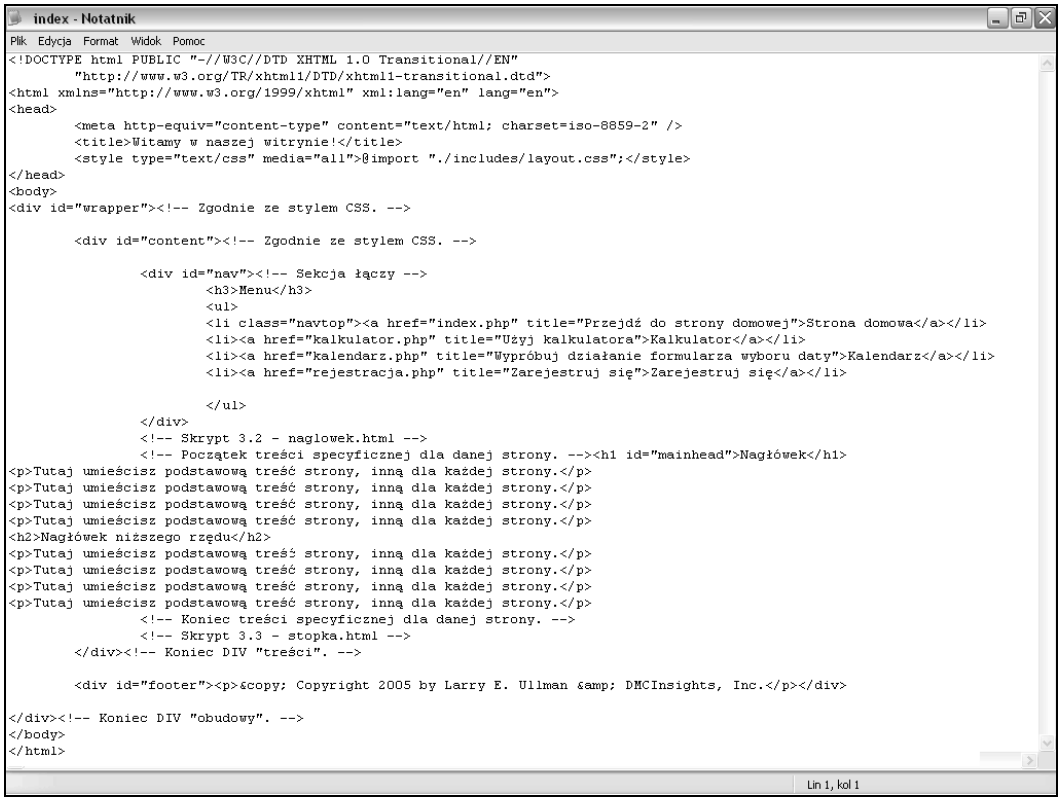
13. Przetestuj szablon ładując *index.php* w przeglądarce internetowej (rysunek 3.4).



Rysunek 3.4. *Tym razem do uzyskania tego samego wyglądu strony (patrz rysunek 3.3) wykorzystałem pliki zewnętrzne PHP*

Strona *index.php* stanowi końcowy rezultat zastosowania systemu szablonu. Nie musimy bezpośrednio odwoływać się do dołączanych plików, ponieważ skrypt *index.php* sam dołączy ich zawartość.

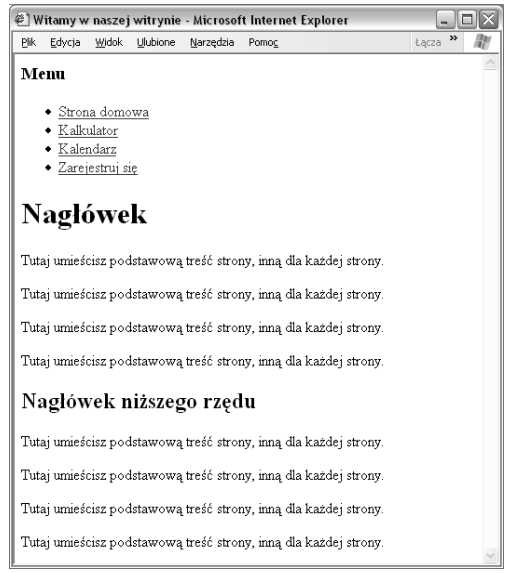
14. Jeśli chcesz, możesz obejrzeć źródło strony HTML (rysunek 3.5).



Rysunek 3.5. *Wygenerowany kod źródłowy w języku HTML powinien być wierną kopią kodu występującego w pierwotnym szablonie stron (patrz listing 3.1)*

Wskazówki

- W pliku *php.ini* występuje ustawienie `include_path`. Dzięki niemu możesz określić, które pliki zewnętrzne mogą być dołączane, a które nie.
- W rozdziale 7., „PHP i MySQL", przekonasz się, że wszystkie pliki zawierające poufne dane (takie jak na przykład informacje potrzebne do połączenia się z bazą danych), powinny być przechowywane w innym katalogu niż pozostałe dokumenty witryny.
- Odwołując się do plików znajdujących się w tym samym katalogu co plik bieżący, powinieneś zawsze stosować składnię `./nazwapliku`. Z kolei do plików znajdujących się w katalogu wyższego poziomu odwołuj się, pisząc `../nazwapliku`, a do plików z katalogu niższego poziomu — pisząc `.katalog/nazwapliku`.
- Dołączając pliki, możesz posługiwać się ścieżkami względnymi (tak jak ja) lub bezwzględnymi: `include ('/ścieżka/do/pliku/nazwapliku');`
- W przypadku gdy nie uda się dołączyć pliku zewnętrznego, funkcja `require()` ma większy wpływ na funkcjonowanie skryptu niż funkcja `include()`. Dlatego też powinno się ją stosować jedynie tam, gdzie dołączenie danego pliku ma kluczowe znaczenie (na przykład wówczas, gdy jest to plik odpowiedzialny za nawiązanie połączenia z bazą danych), a w przypadku dołączania kodu wpływającego jedynie na kosmetykę strony należy posługiwać się funkcją `include()`. Wersje `_once()` przydają się w skomplikowanych aplikacjach, ale w przypadku prostej witryny ich stosowanie nie jest konieczne.
- *CSS* działa w taki sposób, że jeśli nie użyjesz pliku CSS lub nie wczyta go przeglądarka, to strona będzie nadal działać poprawnie, ale jej wygląd będzie mniej estetyczny (patrz rysunek 3.6).



Rysunek 3.6. *Wygląd strony HTML bez użycia pliku CSS (porównaj z rysunkiem 3.4)*

Wyświetlanie i obsługa formularza przez jeden skrypt

We wszystkich dotychczasowych przykładach do obsługi formularzy HTML wykorzystywałem dwa niezależne skrypty. Jeden wyświetlał formularz, a drugi odbierał wysłane z niego dane. Choć nie ma w tym nic złego, czasem wygodnie jest realizować obie te funkcje za pośrednictwem jednego skryptu. Gdy ta sama strona ma jednocześnie wyświetlać i obsługiwać formularz, trzeba zastosować wyrażenie warunkowe.

```
if (/* wysłano dane z formularza */) {
    // Obsłuż je.
} else {
    // Wyświetl formularz.
}
```

Chcąc dowiedzieć się, czy dane z formularza zostały już wysłane, sprawdzam, czy zmiennej `$_POST` przypisano jakąś wartość (zakładając, że formularz używa metody `POST`). Mogę na przykład sprawdzić zmienną `$_POST['wyslany']`, gdzie wysłany jest nazwą ukrytego pola formularza.

```
if (isset($_POST['wyslany'])) {
    // Obsłuż formularz.
} else {
    // Wyświetl formularz.
}
```

Jeśli chcesz, aby skrypt obsłużył formularz, a następnie wyświetlił go ponownie (bo chcesz na przykład dodać kolejny rekord do bazy danych), użyj konstrukcji:

```
if (isset($_POST['wyslany'])) {
    // Obsłuż dane.
}
// Wyświetl formularz.
```

Ta wersja skryptu wyświetla formularz przy każdym załadowaniu strony i obsługuje go, jeżeli zostały z niego wysłane jakieś dane.

Aby zademonstrować tę ważną technikę, stworzę teraz kalkulator sprzedaży. W dalszej części tego rozdziału użyję tej samej metody, tworząc stronę rejestracji.

Aby obsługiwać formularze HTML:

1. Utwórz w edytorze tekstów nowy dokument PHP (listing 3.5).

   ```
   <?php # Skrypt 3.5 - kalkulator.php
   $page_title = 'Kalkulator kosztów';
   include ('./includes/naglowek.html');
   ```

   Ten, i wszystkie pozostałe przykłady w bieżącym rozdziale, będą używać tego samego szablonu co strona *index.php*. Dlatego też składnia początku każdej strony będzie taka sama, różne będą jedynie tytuły.

2. Napisz wyrażenie warunkowe obsługujące formularz.

   ```
   if (isset($_POST['submitted'])) {
   ```

   Jak wcześniej wspomniałem, to, że zmienna `$_POST['submitted']` ma jakąś wartość, oznacza, że dane z formularza zostały już przesłane i można je obsłużyć. Zmienna ta będzie tworzona przez ukryte pole formularza służące tylko jako wskaźnik jego wysłania.

3. Zweryfikuj wprowadzone dane.

   ```
   if ( is_numeric($_POST['quantity'])
     && is_numeric($_POST['price']) &&
     is_numeric($_POST['tax']) ) {
   ```

   Weryfikacja jest w tym przypadku bardzo prosta i polega na sprawdzeniu, czy wartości trzech zmiennych są numeryczne. Można ją rozbudować na przykład o sprawdzenie, czy pierwsza z nich ma wartość całkowitą (w rozdziale 10., „Zabezpieczenia” pojawi się wersja tego skryptu wykonująca taką kontrolę).

   Jeśli weryfikacja zakończy się pomyślnie, obliczenia zostaną wykonane. W przeciwnym razie użytkownik zostanie poproszony o ponowne wprowadzenie danych.

Listing 3.5. *Ten skrypt zarówno wyświetla, jak i obsługuje formularz rejestracyjny*

Listing

```
1   <?php # Skrypt 3.5 - kalkulator.php
2   $page_title = 'Kalkulator kosztów';
3   include
    ('./includes/naglowek.html');
4
5   // Sprawdź czy formularz został wysłany.
6   if (isset($_POST['submitted'])) {
7
8     // Podstawowa weryfikacja danych we formularzu.
9     if ( is_numeric($_POST['quantity']) && is_numeric($_POST['price']) &&
      is_numeric($_POST['tax']) ) {
10
11      // Wylicz wyniki.
12      $taxrate = $_POST['tax'] / 100; // Zamień 5% na .05.
13      $total = ($_POST['quantity'] * $_POST['price']) * ($taxrate + 1);
14
15      // Wyświetl wyniki.
16      echo '<h1 id="mainhead">Całkowity koszt</h1>
17      <p>Kupujesz ' . $_POST['quantity'] . ' sztuk w cenie ' . number_format ($_POST['price'],
        2) . 'zł za egzemplarz. Po uwzględnieniu podatku ' . $_POST['tax'] . '%, daje to
        całkowity koszt ' . number_format ($total, 2) . '.</p><p><br /></p>';
18
19    } else { // Wprowadzono niepoprawne dane.
20      echo '<h1 id="mainhead">Błąd!</h1>
21      <p class="error">Wprowadź poprawną liczbę egzemplarzy, cenę i podatek.</p><p><br
        /></p>';
22    }
23
24  } // Koniec głównego IF.
25
26  // Koniec sekcji PHP i początek formularza HTML.
27  ?>
28  <h2>Kalkulator kosztów</h2>
29  <form action="calculator.php" method="post">
30    <p>Liczba egzemplarzy: <input type="text"  name="quantity" size="5"   maxlength="10"
      /></p>
31    <p>Cena: <input type="text"  name="price" size="5"  maxlength="10" /> </p>
32    <p>Podatek:</b> <input type="text"  name="tax" size="5" maxlength="10" /></p>
33    <p><input type="submit" name="submit" value="Oblicz!" /></p>
34    <input type="hidden" name="submitted" value="TRUE" />
35  </form>
36  <?php
37  include ('./includes/stopka.html');
38  ?>
```

4. Wykonaj obliczenia.

```
$taxrate = $tax / 100;
$total = ($_POST['quantity'] *
$_POST['price']) * ($taxrate + 1);
```

Pierwszy wiersz zamienia wartość wyrażoną w procentach (np. 5%) na ułamek dziesiętny (0.05) używany w dalszych obliczeniach. W drugim wierszu liczba egzemplarzy mnożona jest przez cenę jednostkową. Następnie mnoży się ją przez wartość podatku (0.05) powiększoną o 1 (1.05). To najszybszy sposób wyznaczenia wartości powiększonej o podatek.

5. Wyświetl wyniki.

```
echo '<h1 id="mainhead"
  Całkowity koszt</h1>
  <p>Kupujesz ' . {$_POST
  ['quantity']  . ' sztuk w cenie ' .
  number_format($_POST['price'], 2) .
  ' zł za egzemplarz. Po uwzględnieniu
  podatku' . $_POST['price'] . '% daje
  to ' . number_format($total,2) . '
  zł.</p><p><br /></p>';
```

Wyświetlone są wszystkie wartości, a cena i całkowity koszt zostają dodatkowo sformatowane za pomocą funkcji `number_format()`.

6. Dokończ główne wyrażenie warunkowe i zamknij znacznik PHP.

```
  } else {
    echo '<h1 id="mainhead">Błąd!</h1>
    <p class="error"> Wprowadź poprawną
    liczbę egzemplarzy, cenę i
    podatek.</p><p><br /></p>';
  }
}
?>
```

Klauzula `else` zamyka wyrażenie warunkowe weryfikacji, wyświetlając komunikat o błędzie (jeśli wprowadzone wartości nie są numeryczne). Ostatni nawias klamrowy zamyka natomiast wyrażenie warunkowe `isset($_POST['submitted'])`. Następnie zamykana jest sekcja PHP, dzięki czemu formularz będzie można utworzyć bez stosowania funkcji `print()` lub `echo()`.

7. Wyświetl formularz HTML.

```
<h2>Kalkulator kosztów</h2>
<form action="kalkulator.php"
method="post">
  <p>Liczba egzemplarzy: <input
  type="text" name="quantity" size="5"
  maxlength="10" /></p>
  <p>Cena: <input type="text"
  name="price" size="5" maxlength="10"
/></p>
  <p>Podatek (%): <input type="text"
   name="tax" size="5" maxlength="10"
   /></p>
  <p><input type="submit" name="submit"
  value="Oblicz!" /></p>
  <input type="hidden" name="submitted"
  value="TRUE" />
</form>
```

Formularz jest dość prosty, zawiera jedynie dwie nowe sztuczki. Po pierwsze, atrybut `action` używa nazwy skryptu, dzięki czemu wysłanie formularza powoduje powrót do tej samej strony zamiast przejście do innej. Po drugie, formularz zawiera ukryte pole o nazwie `submitted` i wartości `TRUE`. Pole to spełnia rolę znacznika, którego istnienie jest sprawdzane, aby ustalić, czy formularz wymaga obsługi (patrz główne wyrażenie warunkowe).

8. Dołącz plik stopki.

```
<?php
include ('./includes/stopka.html');
?>
```

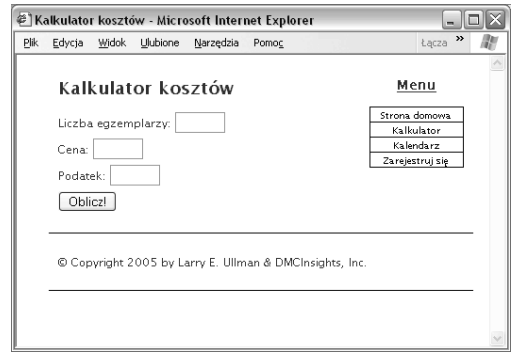
9. Zapisz plik pod nazwą *kalkulator.php*, wgraj go na serwer i przetestuj w przeglądarce internetowej (rysunek 3.7, 3.8 i 3.9).

Wskazówki

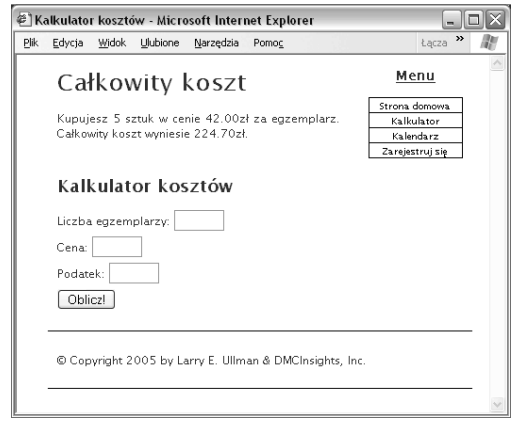
- Aby zbadać, czy dane z formularza zostały już wysłane, możesz to też sprawdzić, określając, czy zmienna przycisku *Wyślij dane* (`$_POST['submit']`) ma wartość. Metoda ta nie zadziała, jeśli użytkownik wyśle formularz używając klawisza *Enter*.
- Jeśli będziesz chciał użyć grafiki jako przycisku wysyłania danych formularza, formularz będzie musiał zawierać ukryte pole umożliwiające sprawdzenie, czy został on wysłany.
- Powrót do tego samego formularza po jego wysłaniu możliwy jest także za pomocą PHP. Można to zrobić, umieszczając nazwę bieżącego skryptu przechowywaną przez zmienną `$_SERVER['PHP_SELF']` jako wartość atrybutu `action`:

```
<form action="<?php echo
$_SERVER['PHP_SELF']; ?> "
method="post">
```

Zaletą tego rozwiązania jest to, że formularz będzie powracać do tej samej strony nawet wtedy, gdy zmienimy później nazwę skryptu.



Rysunek 3.7. *Pierwsze wyświetlenie formularza HTML*



Rysunek 3.8. *Strona wykonuje obliczenia, prezentuje wyniki i ponownie wyświetla formularz*

Rysunek 3.9. *Jeżeli jedna z wprowadzonych wartości nie jest numeryczna, zostaje wyświetlony komunikat o błędzie*

Tworzenie formularzy z pamięcią

Na pewno zetknąłeś się już z *formularzami z pamięcią*, nawet jeśli nie wiesz, że właśnie tak się je nazywa. Są to zwykłe formularze HTML, które zapamiętują wszystko, co do nich wpisałeś. Z punktu widzenia użytkownika końcowego jest to bardzo przydatne, zwłaszcza gdy trzeba ponownie wypełnić ten sam formularz.

Aby określić początkową wartość widniejącą w tekstowym polu wejściowym, wprowadź atrybut `value`:

```
<input type="text" name="city" size="20"
  value="Innsbruck" />
```

Jeśli chcesz, aby to PHP określił tę wartość, użyj po prostu polecenia `echo()` dla odpowiedniej zmiennej:

```
<input type="text" name="city" size="20"
  value="<?php echo $city; ?>" />
```

Zmodyfikuję teraz skrypt *kalkulator.php* w taki sposób, aby zapamiętywał on wprowadzane dane.

Aby utworzyć formularz z pamięcią:

1. Otwórz w edytorze tekstów plik *kalkulator.php* (patrz listing 3.5).
2. Zmień definicję pola wejściowego `quantity` (listing 3.6):

```
<p>Nazwisko: <input type="text"
  name="quantity" size="5"
  maxlength="10" value="<?php if
 (isset($_POST['quantity'])) echo
  $_POST['quantity']; ?>" /></p>
```

   Po pierwsze, dodałem do pola wejściowego atrybut `value`. Następnie przesłałem do przeglądarki wartość zmiennej `$_POST['quantity']`. Muszę jednak najpierw upewnić się, czy ma ona w ogóle jakąś wartość. W tym celu zastosowałem następujący fragment kodu:

```
<?php
if (isset($_POST['quantity'])) {
    echo $_POST['quantity'];
}
?>
```

   Kod ten zapisałem w skrypcie maksymalnie zwięźle (korzystając z tego, że jeśli blok wyrażenia warunkowego zawiera tylko jedną instrukcję, to można pominąć nawiasy klamrowe).

3. Powtórz te operacje dla ceny i podatku.

```
<p>Cena: <input type="text"
  name="price" size="5"
  maxlength="10" value="<?php if
  (isset($_POST['price'])) echo
  $_POST['price']; ?>" /> </p>
<p>Podatek:</b> <input type="text"
  name="tax" size="5" maxlength="10"
  value="<?php if (isset($_POST
  ['tax'])) echo $_POST['tax'];
?>" /></p>
```

Listing 3.6. *Formularz kalkulatora zapamiętuje teraz informacje wprowadzane przez użytkownika*

Listing

```
<?php # Script 3.6 - calculator.php
$page_title = 'Kalkulator kosztów';
include ('./includes/naglowek.html');

// Sprawdź czy formularz został wysłany.
if (isset($_POST['submitted'])) {

// Podstawowa weryfikacja danych we formularzu.
if ( is_numeric($_POST['quantity']) && is_numeric($_POST['price']) &&
is_numeric($_POST['tax']) ) {

  // Wylicz wyniki.
  $taxrate = $_POST['tax'] / 100; // Zamień 5% na .05.
  $total = ($_POST['quantity'] * $_POST['price']) * ($taxrate + 1);

  // Wyświetl wyniki.
  echo '<h1 id="mainhead">Całkowity koszt</h1>
  <p>Kupujesz ' . $_POST['quantity'] . ' sztuk w cenie ' . number_format ($_POST['price'],
2)
  . 'zł za egzemplarz. Po uwzględnieniu podatku ' . $_POST['tax'] . '%, daje to całkowity
  koszt ' . number_format ($total, 2) . '.</p><p><br /></p>';

  } else { // Wprowadzono niepoprawne dane.
      echo '<h1 id="mainhead">Błąd!</h1>
      <p class="error">Wprowadź poprawną liczbę egzemplarzy, cenę i podatek.</p><p><br
      /></p>';
  }

} // Koniec głównego IF.

// Koniec sekcji PHP i początek formularza HTML.
?>
<h2>Kalkulator kosztów</h2>
<form action="calculator.php" method="post">
  <p>Liczba egzemplarzy: <input type="text"  name="quantity" size="5"   maxlength="10"
  value="<?php if (isset($_POST['quantity'])) echo  $_POST['quantity']; ?>" /></p>
  <p>Cena: <input type="text"  name="price" size="5"  maxlength="10" value="<?php if
  (isset($_POST['price'])) echo  $_POST['price']; ?>" /> </p>
  <p>Podatek:</b> <input type="text"  name="tax" size="5" maxlength="10"  value="<?php if
  (isset($_POST  ['tax'])) echo $_POST['tax'];?>" /></p>
  <p><input type="submit" name="submit" value="Oblicz!" /></p>
  <input type="hidden" name="submitted" value="TRUE" />
</form>
<?php
include ('./includes/stopka.html');
?>
```

Rysunek 3.10. *Ta wersja formularza zapamiętuje informacje wprowadzone poprzednim razem…*

Rysunek 3.11. *… niezależnie od tego czy formularz został wypełniony w całości.*

- Aby wstępnie wybrać pozycję menu, użyj atrybutu `selected`:

```
<select name="year">
<option value= "2005">2005</option>
<option value= "2006"
selected="selected">2006</option>
</select>
```

  Przykład takiego rozwiązania pokażę pod koniec rozdziału.

4. Zapisz plik pod nazwą *kalkulator.php*, wgraj go na serwer i przetestuj w przeglądarce internetowej (rysunek 3.10 i 3.11).

Wskazówki

- Ponieważ w tym przykładzie kod PHP występuje również jako wartość atrybutu HTML o nazwie `value`, to komunikaty o błędach mogą wydawać się niezrozumiałe. W przypadku takich problemów, sprawdź w kodzie źródłowym HTML, czy błędy PHP są wyświetlane wewnątrz atrybutu `value`.
- Wartości atrybutów HTML powinieneś zawsze umieszczać w cudzysłowach. Dotyczy to zwłaszcza atrybutu `value`. W przeciwnym razie wartość złożona z kilku słów, na przykład *Jan Kowalski*, pojawi się w przeglądarce jako *Jan*.
- Jeśli serwer używa opcji *Magic Quotes*, to zanim wyświetlisz zmienne łańcuchowe jako wartości pól wejściowych, powinieneś zastosować funkcję `stripslashes()`:

```
<input type="text" name="last_name"
  size="20" value="<?php if (isset
 ($_POST['last_name'])) echo
  stripslashes($_POST['last_name']);
?>" />
```

- Ze względu na ograniczenia wynikające ze sposobu działania języka HTML nie można określić początkowej wartości pola wejściowego typu `password`.
- Aby wstępnie zaznaczyć wybrane przyciski opcji i pola wyboru, wprowadź do definiujących je znaczników atrybut `checked="checked"`.

```
<input type="checkbox"
  name="interests" value="Narty"
  checked="checked" />
<input type="radio"
  name="gender" value="Kobieta"
  checked="checked" />
```

- Aby określić początkową zawartość obszaru tekstowego, umieść ją między znacznikami `textarea`:

```
<textarea name="comments"
  rows="10" cols="50"><?php echo
  $_POST['comments']; ?></textarea>
```

Tworzenie i wywoływanie własnych funkcji

Jak zdążyłeś się już przekonać, PHP ma bardzo wiele wbudowanych funkcji zdolnych zaspokoić niemal każde potrzeby. Ważniejsze jest jednak to, że możesz tworzyć własne funkcje realizujące określone przez Ciebie zadania. Tworząc funkcje, posługujemy się następującą składnią:

```
function nazwa_funkcji() {
  // Kod funkcji.
}
```

Nazwa Twojej funkcji może być dowolną kombinacją liter, cyfr i znaków podkreślenia, przy czym nie może zaczynać się od cyfry. Nie możesz również używać nazw funkcji, które już istnieją (`print`, `echo`, `isset` itp.).

W rozdziale 1. wspominałem już, że PHP nie rozróżnia w nazwach funkcji małych i wielkich liter (zatem inaczej niż w przypadku zmiennych), w związku z czym zdefiniowaną powyżej funkcję można wywołać, pisząc: `nazwa_funkcji()`, `nazwa_Funkcji()`, `NAZWA_FUNKCJI()` itp.

Kod występujący w ciele funkcji może wykonywać niemal dowolne operacje, począwszy od generowania kodu HTML, na przeprowadzaniu obliczeń skończywszy. Właśnie takie czynności będą wykonywały funkcje, które zdefiniuję w tym rozdziale.

Aby utworzyć własną funkcję:

1. Utwórz w edytorze tekstów nowy dokument PHP (listing 3.7).

   ```
   <?php # Skrypt 3.7 - formularzdaty.php
   $page_title = 'Formularz Kalendarza';
   include ('./includes/naglowek.html');
   ```

   Strona ta będzie wykorzystywała ten sam szablon co dwie poprzednie.

Listing 3.7. *Ta funkcja przydaje się przy tworzeniu większej liczby menu rozwijalnych*

Listing

```
 1  <?php # Skrypt 3.7 - kalendarz.php
 2  $page_title = 'Kalendarz';
 3  include ('./includes/naglowek.html');
 4
 5  // Funkcja tworząca trzy menu rozwijalne
    wyboru miesięcy, dni i lat.
 6  function make_calendar_pulldowns() {
 7
 8    // Utwórz tablicę miesięcy.
 9    $months = array (1 => 'Styczeń', 'Luty',
     'Marzec', 'Kwiecień', 'Maj', 'Czerwiec',
     'Lipiec', 'Sierpień', 'Wrzesień',
     'Październik', 'Listopad', 'Grudzień');
10
11    // Utwórz menu miesięcy.
12    echo '<select name="month">';
13    foreach ($months as $key => $value) {
14      echo "<option value=\"$key\
        ">$value</option>\n";
15    }
16    echo '</select>';
17
18    // Utwórz menu dni.
19    echo '<select name="day">';
20    for ($day = 1; $day <= 31; $day++) {
21      echo "<option value=\"$day\">
        $day</option>\n";
22    }
23    echo '</select>';
24
25    // Utwórz menu lat.
26    echo '<select name="year">';
27    for ($year = 2005; $year <= 2015; $year++) {
28      echo "<option value=\"$year\
        ">$year</option>\n";
29    }
30    echo '</select>';
31
32  } // Koniec definicji funkcji.
33
34  // Utwórz znaczniki formularza
35  echo '<h1 id="mainhead">Wybierz datę:</h1>
36  <p><br /></p><form action="kalendarz.php"
    method="post">';
37
38  // Wywołaj funkcję.
39  make_calendar_pulldowns();
40
41  echo '</form><p><br /></p>'; // Koniec
    formularza.
42
43  include ('./includes/stopka.html');
44  ?>
```

2. Zacznij od zdefiniowania nowej funkcji.

```
function make_calendar_pulldowns() {
```

Funkcja, którą napiszę, będzie generowała menu rozwijalne dla dni, miesięcy i lat, podobnie jak skrypt *kalendarz.php* (patrz listing 2.12). Nazwa, którą nadałem funkcji, jednoznacznie określa jej przeznaczenie[1].

Chociaż nie jest to wymagane, definicje funkcji umieszcza się zwykle na początku skryptu lub w osobnym pliku.

3. Wygeneruj menu rozwijalne.

```
$months = array (1 => 'Styczeń', 'Luty',
'Marzec', 'Kwiecień', 'Maj', 'Czerwiec',
'Lipiec', 'Sierpień', 'Wrzesień',
'Październik', 'Listopad', 'Grudzień');
echo <select name="month">';
foreach ($months as $key => $value) {
  echo "<option value=\"$key\">
    $value</option>\n";
}
echo '</select>
echo'<select name="day">';
for ($day = 1; $day <= 31; $day++) {
  echo "<option value=\"$day\">
    $day</option>\n";
}
echo '</select>
<select name="year">';
for ($year=2003; $year <= 2010;
  $year++) {
  echo "<option value=\"$year\">
    $year</option>\n";
  }
echo '</select>';
```

Jest to dokładnie ten sam kod co w oryginalnym skrypcie, z tym że teraz umieściliśmy go w ciele funkcji. Jedyna drobna zmiana polega na użyciu pętli `for` do stworzenia menu rozwijalnego lat (poprzednio wykorzystaliśmy w tym celu pętlę `while`).

4. Zamknij definicję funkcji.

```
} // Koniec definicji funkcji.
```

Gdy kod jest dość skomplikowany, warto wstawiać na końcu definicji funkcji komentarze, ponieważ dzięki nim wiadomo, gdzie definicja się kończy, a gdzie zaczyna.

[1] *make callendar pulldown* to po angielsku „utwórz menu rozwijalne kalendarza” — *przyp. tłum.*

5. Utwórz formularz i wywołaj funkcję.

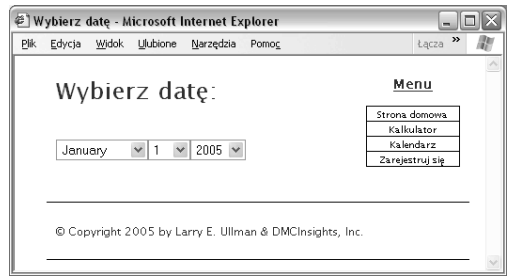
```
echo '<h1 id="mainhead"
  Wybierz datę:</h1>
  <p><br /></p><form action=
  "formularzdaty.php" method="post">';
make_calendar_pulldowns();
echo '</form><p><br /></p>';
```

Ten fragment kodu tworzy znaczniki definiujące formularz (oraz używa znaczników HTML do stworzenia układu strony) i wywołuje funkcję `make_calendar_pulldown()`, która generuje kod opisujący trzy menu rozwijalne.

6. Dokończ skrypt PHP, dołączając plik stopki.

```
include ('./includes/stopka.html');
```

7. Zapisz plik pod nazwą *formularzdaty.php*, wgraj go na serwer (do tego samego katalogu co plik *index.php*) i przetestuj w przeglądarce internetowej (rysunek 3.12).



Rysunek 3.12. *Menu rozwijalne wygenerowane przez funkcję zdefiniowaną przez użytkownika*

Wskazówki

- Jeśli pojawi się komunikat o błędzie *call to undefined function* `nazwa_funkcji`, oznacza on, że wywołałeś funkcję, która nie jest zdefiniowana. Może się to zdarzyć na skutek błędnego wpisania nazwy funkcji (podczas jej definiowania lub wywołania) lub gdy zapomnisz dołączyć plik, w którym zdefiniowałeś funkcję.
- Ponieważ funkcje definiowane przez użytkownika zajmują nieco miejsca w pamięci, powinieneś zawsze zastanowić się, czy w danym przypadku warto je stosować. Zazwyczaj ich użycie opłaca się wówczas, gdy dany fragment kodu wykonywany jest wiele razy w różnych miejscach skryptu lub witryny.

Rysunek 3.13. *Często popełnianym błędem jest pominięcie któregoś z argumentów lub przekazanie do funkcji argumentu niewłaściwego typu*

Tworzenie funkcji pobierających argumenty

Funkcje tworzone przez użytkownika, podobnie jak wbudowane funkcje PHP, mogą pobierać argumenty (zwane także parametrami). Na przykład argumentem funkcji `print()` jest to, co chcesz przesłać do przeglądarki, a argumentem `strlen()` jest łańcuch znaków, którego długość chcesz poznać.

Każda funkcja może pobierać dowolną liczbę argumentów, przy czym niezwykle istotna jest kolejność ich występowania. Definiując funkcję, wprowadź w miejscu występowania argumentów nazwy zmiennych:

```
function print_hello ($first, $last) {
  // Kod funkcji.
}
```

Funkcję tę możesz następnie wywołać jak każdą inną funkcję PHP, przekazując jej zmienne lub literały:

```
print_hello ('Jimmy', 'Stewart');
$surname = 'Stewart';
print_hello ('Jimmy', $surname);
```

Tak samo jak w przypadku każdej innej funkcji, podanie nieprawidłowej liczby argumentów, spowoduje wystąpienie błędu (rysunek 3.13). Nazwy zmiennych, które wybierzesz dla argumentów, nie mają znaczenia z punktu widzenia pozostałej części skryptu (dowiesz się o tym w podrozdziale „Zasięg zmiennej"), dlatego postaraj się, aby były one jednoznaczne i jak najbardziej opisowe.

Aby zademonstrować omawiane tu zagadnienia, przepiszę teraz przykład z kalkulatorem, tak aby wykorzystywał on funkcję.

Aby zdefiniować funkcje pobierające argumenty:

1. Otwórz w edytorze tekstów skrypt *kalkulator.php* (listing 3.8).

2. Tuż za wierszem, w którym dołączasz plik nagłówka, zdefiniuj funkcję `calculate_total()`.

```
function calculate_total ($qty,
  $cost, $tax) {
  $taxrate = $tax / 100;
  $total = ($qty * $cost) *
    ($taxrate + 1);
  $total = number_format ($total, 2);
  echo '<p>Kupujesz ' . $qty . '
    sztuk w cenie' . number_format
    ($cost, 2) . ' zł za egzemplarz.
    Po uwzględnieniu podatku ' . $tax .
    '% daje to' . number_format
    ($total, 2) .  'zł.</p>";
}
```

Funkcja ta przeprowadza te same obliczenia co w przykładzie z rozdziału 2. i zwraca wyniki. Pobiera ona trzy argumenty: liczbę zamówionych sztuk, cenę jednostkową i wysokość podatku. Zwróć uwagę, że zmiennymi używanymi przez funkcję nie są `$_POST['quantity']`, `$_POST['price']` ani `$_POST['tax']`. Zmienne argumentów funkcji posiadają swoje własne nazwy ważne w obrębie danej funkcji.

Listing 3.8. *Ta wersja skryptu kalkulatora używa do wykonania obliczeń funkcji z parametrami*

Listing

```
1  <?php # Skrypt 3.8 - kalkulator.php
   (trzecia wersja skryptów 3.5 i 3.6)
2  $page_title = 'Kalkulator kosztów';
3  include ('./includes/naglowek.html');
4
5  /* Funkcja wyliczająca całkowity koszt
6  i wyświetlająca wynik. */
7  function calculate_total ($qty, $cost,
   $tax) {
8
9  $taxrate = $tax / 100; // Zamień 5%
   na .05.
10 $total = ($qty * $cost) * ($taxrate + 1);
11 echo '<p>Kupujesz ' . $qty . ' sztuk
   w cenie ' . number_format($cost, 2) .
   'zł za egzemplarz. Po uwzględnieniu
   podatku ' . $tax .     '% daje to
   całkowity koszt ' . number_
   format($total, 2) .  'zł.</p>';
12
13 } // Koniec funkcji.
14
15 // Sprawdź czy formularz został
   wysłany.
16 if (isset($_POST['submitted'])) {
17
18 // Podstawowa weryfikacja danych
   formularza.
19 if ( is_numeric($_POST['quantity']) &&
   is_numeric($_POST['price']) &&
   is_numeric($_POST['tax']) ) {
20
21 // Wyświetl nagłówek.
22 echo '<h1 id="mainhead">Całkowity
   koszt</h1>';
23
24 // Wywołaj funkcję.
25 calculate_total ($_POST['quantity'],
   $_POST['price'], $_POST['tax']);
26
```

Listing 3.8. *Ta wersja skryptu kalkulatora używa do wykonania obliczeń funkcji z parametrami — ciąg dalszy*

Listing

```
27  // Wyświetl odstępy
28  echo '<p><br /></p>';
29
30  } else { // Wprowadzono niepoprawne
    wartości.
31      echo '<h1 id="mainhead">Błąd!</h1>
32      <p class="error">Wprowadź poprawną
        liczbę egzemplarzy, cenę i
        podatek.</p><p><br /></p>';
33  }
34
35  } // Koniec głównego IF.
36
37  // Koniec sekcji PHP i początek
    formularza HTML.
38  ?>
39  <h2>Kalkulator kosztów</h2>
40  <form action="kalkulator.php"
    method="post">
41  <p>Liczba egzemplarzy: <input
    type="text"  name="quantity" size="5"
    maxlength="10" value="<?php if
   (isset($_POST['quantity'])) echo
    $_POST['quantity']; ?>" /></p>
42  <p>Cena: <input type="text"
    name="price" size="5"  maxlength="10"
    value="<?php if  (isset($_POST[
   'price'])) echo  $_POST['price']; ?>" />
    </p>
43  <p>Podatek:</b> <input type="text"
    name="tax" size="5" maxlength="10"
    value="<?php if (isset($_POST  ['tax']))
    echo $_POST['tax'];?>" /></p>
44  <p><input type="submit" name="submit"
    value="Oblicz!" /></p>
45  <input type="hidden" name="submitted"
    value="TRUE" />
46  </form>
47  <?php
48  include ('./includes/stopka.html');
49  ?>
```

3. Zmień wyrażenie warunkowe strony weryfikujące dane (poprzednio były w nim wykonywane obliczenia).

```
echo '<h1 id="mainhead">Ogólny koszt</h1>
calculate_total ($_POST['quantity'],
  $_POST['price'], $_POST['tax']);
echo '<p><br /><p>';
```

Również i tym razem tylko nieznacznie zmieniłeś sposób działania skryptu. Wyświetlany jest nagłówek, a następnie wywoływana funkcja i dodawane odpowiednie odstępy na stronie.

Wywoływanej funkcji zostają przekazane trzy argumenty, z których każdy jest zmienną `$_POST`. Wartość zmiennej `$_POST['quantity']` zostaje przypisana zmiennej `$qty` funkcji; wartość zmiennej `$_POST['price']` zostaje przypisana zmiennej `$cost` funkcji; wartość zmiennej `$_POST['tax']` zostaje przypisana zmiennej `$tax` funkcji.

4. Zapisz plik pod nazwą *kalkulator.php*, wgraj go na serwer i przetestuj w przeglądarce internetowej (rysunek 3.14).

Rysunek 3.14. *Obliczenia wykonuje teraz funkcja zdefiniowana przez użytkownika, która wyświetla również wyniki*

Nadawanie argumentom wartości domyślnych

Własne funkcje można też definiować w nieco inny sposób, a mianowicie ustalając domyślne wartości argumentów.

```
function greet($name, $greeting =
  'Hello') {
  echo "$greeting, $name!";
}
```

Na skutek określenia wartości domyślnej dla danego argumentu staje się on argumentem opcjonalnym i nie trzeba umieszczać go w wywołaniu funkcji. Jeżeli to zrobisz, wykorzystana zostanie podana przez Ciebie wartość. Jeśli nie, w funkcji będzie obowiązywała wartość domyślna.

Możesz określić wartości domyślne dla dowolnej liczby argumentów, pod warunkiem że są to ostatnie argumenty na liście parametrów funkcji. Innymi słowy, wszystkie te argumenty, których wartość musi być określona przez użytkownika, muszą znaleźć się na początku listy argumentów.

W przypadku funkcji zdefiniowanej powyżej wszystkie pokazane tu wywołania są prawidłowe:

```
greet($surname, $message);
greet('Roberts');
greet('Grant', 'Dobry wieczór');
```

Natomiast wywołanie `greet()` jest nieprawidłowe, ponieważ nie można nadać wartości argumentowi `$greeting` bez nadania wartości argumentowi `$name`.

Aby określić domyślne wartości argumentów:

1. Otwórz w edytorze tekstów plik *kalkulator.php* (patrz listing 3.8).
2. Zmień wiersz zawierający definicję funkcji (wiersz 7.) w taki sposób, aby wymaganymi argumentami były liczba sztuk i cena jednostkowa (`$qty` i `$cost`, listing 3.9).

   ```
   function calculate_total
   ($qty, $cost, $tax = 5) {
   ```

   Wartość zmiennej `$tax` jest teraz zakodowana na stałe w definicji funkcji a tym samym jest opcjonalna.

Listing 3.9. *Jeżeli w wywołaniu tej wersji funkcji calculate_total() nie zostanie określona wartość podatku, to funkcja wykorzysta wartość domyślną*

Listing

```
1  <?php # Skrypt 3.9 - kalkulator.php
   (czwarta wersja skryptów 3.5, 3.6 i 3.8)
2  $page_title = 'Kalkulator kosztów';
3  include ('./includes/naglowek.html');
4
5  /* Funkcja wyliczająca całkowity koszt
6  i wyświetlająca wynik. */
7  function calculate_total ($qty, $cost,
   $tax = 5) {
8
9    $taxrate = $tax / 100; // Zamień 5% na
     .05.
10   $total = ($qty * $cost)
     * ($taxrate + 1);
11   echo '<p>Kupujesz ' . $qty . ' sztuk w
     cenie ' . number_format($cost, 2) . 'zł
     za egzemplarz. Po uwzględnieniu podatku
     ' . $tax .     '% daje to całkowity
     koszt ' . number_format($total, 2) .
     'zł.</p>';
12
13 } // Koniec funkcji.
14
15 // Sprawdź czy formularz został wysłany.
16 if (isset($_POST['submitted'])) {
17
18   if (is_numeric($_POST['quantity']) &&
     is_numeric($_POST['price'])) {
19
20     // Wyświetl nagłówek.
21     echo '<h1 id="mainhead">Całkowity
       koszt</h1>';
22
23     if (is_numeric($_POST['tax'])) {
24       calculate_total
         ($_POST['quantity'],
         $_POST['price'], $_POST['tax']);
25     } else {
26       calculate_total
         ($_POST['quantity'],
         $_POST['price']);
27     }
28
29   // Wyświetl odstępy.
30   echo '<p><br /></p>';
31
32   } else { // Wprowadzono niepoprawne
     wartości.
```

Listing 3.9. *Jeżeli w wywołaniu tej wersji funkcji calculate_total() nie zostanie określona wartość podatku, to funkcja wykorzysta wartość domyślną — ciąg dalszy*

Listing

```
33        echo '<h1 id="mainhead">Błąd!</h1>
34         <p class="error">Wprowadź poprawną
           liczbę egzemplarzy i cenę
           jednostkową.</p><p><br /></p>';
35     }
36
37  } // Koniec głównego IF.
38
39  // Koniec sekcji PHP i początek formularza
    HTML.
40  ?>
41  <h2>Kalkulator kosztów</h2>
42  <form action="kalkulator.php"
    method="post">
43     <p>Liczba egzemplarzy: <input
       type="text"  name="quantity" size="5"
       maxlength="10" value="<?php if
       (isset($_POST['quantity'])) echo
       $_POST['quantity']; ?>" /></p>
44     <p>Cena: <input type="text"
       name="price" size="5"  maxlength="10"
       value="<?php if  (isset($_POST
       ['price'])) echo  $_POST['price'];
       ?>" /> </p>
45     <p>Podatek:</b> <input type="text"
       name="tax" size="5" maxlength="10"
       value="<?php if (isset($_POST
       ['tax'])) echo $_POST['tax'];?>"
       />(opcjonalny)</p>
46     <p><input type="submit" name="submit"
       value="Oblicz!" /></p>
47     <input type="hidden" name="submitted"
       value="TRUE" />
48  </form>
49  <?php
50  include ('./includes/stopka.html');
51  ?>
```

3. Zmień weryfikację danych formularza na

```
if (is_numeric($_POST['quantity'])
  && is_numeric($_POST['price'])) {
```

Ponieważ wysokość podatku jest opcjonalna, weryfikuję tylko pierwsze dwie zmienne.

4. Zmień sposób wywołania funkcji:

```
if (is_numeric($_POST['tax'])) {
  calculate_total ($_POST
    ['quantity'], $_POST['price'],
    $_POST['tax']);
} else {
  calculate_total ($_POST
    ['quantity'], $_POST['price']);
}
```

Jeśli użytkownik wprowadził wartość podatku i jest ona liczbą, to funkcja zostanie wywołana w taki sam sposób jak dotychczas. W przeciwnym razie funkcja otrzyma tylko dwa pierwsze argumenty, a dla podatku użyje wartości domyślnej.

5. Zmień komunikat o błędzie, aby dotyczył jedynie liczby sztuk i kosztu jednostkowego.

```
echo '<h1 id="mainhead">Błąd!</h1>
  <p class="error">Proszę wprowadź
  poprawną liczbę sztuk i cenę
  jednostkową.</p><p><br /></p>';
```

Ponieważ wartość podatku jest teraz opcjonalna, to komunikat został odpowiednio zmieniony.

6. Możesz również zaznaczyć w formularzu, że wartość podatku jest opcjonalna.

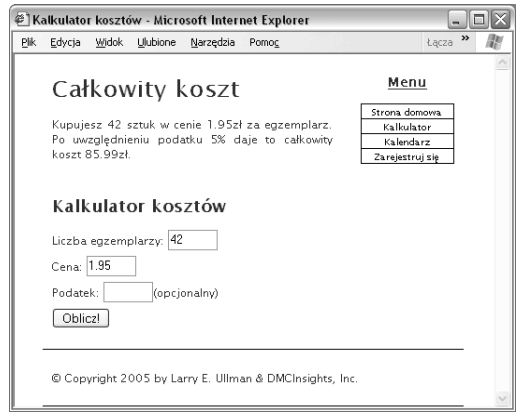
```
<p>Podatek (%): <input type="text"
  name="tax" size="5" maxlength="10"
  value="<?php if (isset($_POST
  ['tax'])) echo $_POST['tax']; ?>"
  /> (opcjonalny)</p>
```

Pole wejściowe zostało opatrzone stosownym komentarzem w nawiasie.

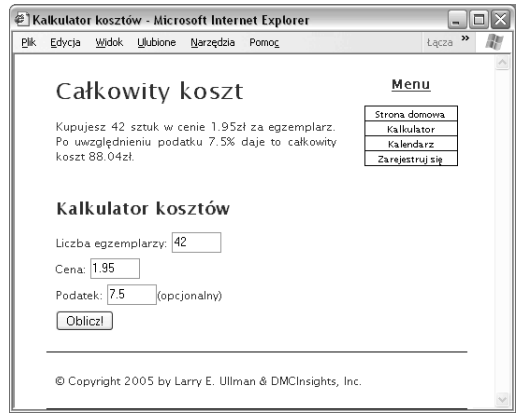
7. Zapisz plik, wgraj go na serwer i przetestuj w przeglądarce internetowej (rysunek 3.15 i 3.16).

Wskazówki

- Jeżeli nie chcesz nadawać określonemu parametrowi funkcji żadnej wartości, przekaż do niego łańcuch pusty (`''`) lub wartość `NULL` bądź `FALSE`.
- W podręczniku PHP opcjonalne parametry funkcji oznaczone są nawiasami kwadratowymi (`[]`).



Rysunek 3.15. *Jeśli użytkownik nie wprowadzi wartości podatku, to skrypt użyje domyślnej wartości 5%*



Rysunek 3.16. *Jeśli użytkownik wprowadzi wartość podatku, to skrypt użyje jej zamiast wartości domyślnej*

Listing 3.10. *Funkcja calculate_total() pobiera teraz do trzech argumentów i zwraca wynik*

Listing

```
1   <?php # Skrypt 3.10 - kalkulator.php
    (piąta wersja skryptów 3.5, 3.6, 3.8 i 3.9)
2   $page_title = 'Kalkulator kosztów';
3   include ('./includes/naglowek.html');
4
5   /* Funkcja wyliczająca całkowity koszt
6   i zwracająca wynik.
7   function calculate_total ($qty, $cost,
    $tax = 5) {
8
9     $taxrate = $tax / 100; // Zamień 5%
      na .05.
10    $total = ($qty * $cost) * ($taxrate
      + 1);
11    return number_format ($total, 2);
12
13  } // Koniec funkcji.
14
15  // Sprawdź czy formularz został
    wysłany.
16  if (isset($_POST['submitted'])) {
17
18    if (is_numeric($_POST['quantity'])
      && is_numeric($_POST['price'])) {
19
20      // Wyświetl nagłówek.
21      echo '<h1 id="mainhead">Całkowity
        koszt</h1>';
22
23      if (is_numeric($_POST['tax'])) {
24          $total_cost = calculate_total
            ($_POST['quantity'],
             $_POST['price'],
             $_POST['tax']);
25      } else {
26          $total_cost = calculate_total
            ($_POST['quantity'],
             $_POST['price']);
27      }
28
29          echo '<p>Kupujesz ' .
            $_POST['quantity'] . ' sztuk
            w cenie ' . number_
            format($_POST['price'], 2) .
            'zł za egzemplarz. Całkowity
             koszt wyniesie ' .
             $total_cost .  'zł.</p>';
```

Zwracanie wartości z funkcji

Ostatnią cechą funkcji, o której powinienem wspomnieć, jest możliwość zwracania wyników (wartości). Robią to niektóre funkcje, choć nie wszystkie. Na przykład, funkcja `print()` zwraca wartość `0` lub `1`, informując, czy została ona wykonana bez przeszkód, natomiast `echo()` nie dostarcza żadnych informacji zwrotnych. Funkcja `strlen()` zwraca liczbę znaków w łańcuchu.

Jeśli chcesz, aby Twoja funkcja zwracała jakąś wartość, umieść w niej wyrażenie `return`.

```
function find_sign ($month, $day) {
    // Kod funkcji.
    return $sign;
}
```

Funkcja może zwrócić wartość (na przykład łańcuch znaków lub liczbę) lub zmienną, której przypisała wcześniej jakąś wartość. Wywołując przykładową funkcję, możemy przypisać zwracaną przez nią wartość do jakiejś zmiennej:

```
$my_sign = find_sign ('October', 23);
```

lub użyć jej jako parametru innej funkcji:

```
print find_sign ('October', 23);
```

Aby funkcja zwracała wartość:

1. Otwórz w edytorze tekstów plik *kalkulator.php* (patrz listing 3.9).
2. Zmień definicję funkcji (listing 3.10) na:

```
function calculate_total ($qty,
  $cost, $tax = 5) {
  $taxrate = $tax / 100;
  $total = ($qty * $cost) *
    ($taxrate + 1);
  return number_format ($total, 2);
}
```

Ta wersja funkcji zwraca jedynie obliczoną sumę bez żadnych znaczników HTML-a i bez przesyłania czegokolwiek do przeglądarki internetowej.

3. Zmień sposób wywołania funkcji na:

```
if (is_numeric($_POST['tax'])) {
  $total_cost = calculate_total
   ($_POST['quantity'], $_POST
    ['price'], $_POST['tax']);
} else {
  $total_cost = calculate_total
   ($_POST['quantity'], $_POST
   ['price']);
}
```

Zmienna `$total_cost` otrzymuje teraz wartość zwracaną przez funkcję `calculate_total()`.

4. Dodaj nowe wyrażenie `echo()` wyświetlające wynik.

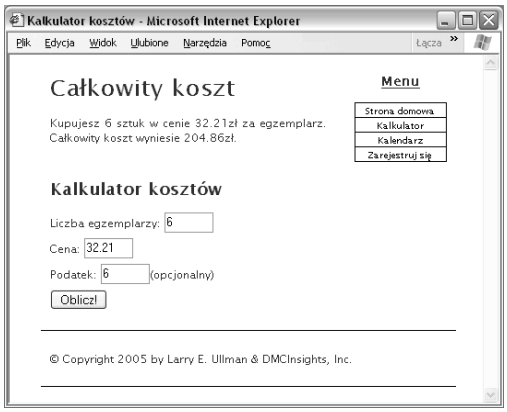
```
echo '<p>Kupujesz ' . {$_POST
  ['quantity']  . ' sztuk w cenie ' .
  number_format($_POST['price'], 2) .
  ' zł za egzemplarz. Po uwzględnieniu
  podatku daje to ' . $total_cost '
  zł.</p>';
```

Ponieważ funkcja zwraca jedynie wynik, to musimy dodać wyrażenie `echo()`, które go wyświetli.

Listing 3.10. *Funkcja calculate_total() pobiera teraz do trzech argumentów i zwraca wynik — ciąg dalszy*

Listing

```
30
31      // Wyświetl odstępy.
32      echo '<p><br /></p>';
33
34      } else { // Wprowadzono niepoprawne
         wartości.
35          echo '<h1 id="mainhead">Błąd!</h1>
36          <p class="error">Wprowadź poprawną
            liczbę egzemplarzy i cenę
            jednostkową.</p><p><br /></p>';
37  }
38
39  } // Koniec głównego IF.
40
41  // Koniec sekcji PHP i początek formularza
    HTML.
42  ?>
43  <h2>Kalkulator kosztów</h2>
44  <form action="kalkulator.php"
    method="post">
45    <p>Liczba egzemplarzy: <input
      type="text"  name="quantity" size="5"
      maxlength="10" value="<?php if
      (isset($_POST['quantity'])) echo
      $_POST['quantity']; ?>" /></p>
46    <p>Cena: <input type="text"
      name="price" size="5"  maxlength="10"
      value="<?php if
      (isset($_POST['price'])) echo
      $_POST['price']; ?>" /> </p>
47    <p>Podatek:</b> <input type="text"
      name="tax" size="5" maxlength="10"
      value="<?php if (isset($_POST  ['tax']))
      echo $_POST['tax'];?>" />(opcjonalny)</p>
48    <p><input type="submit" name="submit"
      value="Oblicz!" /></p>
49    <input type="hidden" name="submitted"
      value="TRUE" />
50  </form>
51  <?php
52  include ('./includes/stopka.html');
53  ?>
```



Rysunek 3.17. *Cel stawiany przed skryptem można zrealizować na wiele różnych sposobów, a efekt będzie za każdym razem taki sam*

5. Zapisz plik, wgraj go na serwer i przetestuj w przeglądarce internetowej (rysunek 3.17).

Wskazówki

- Chociaż ostatni przykład może wydawać się bardziej skomplikowany (funkcja wykonuje tylko obliczenia, a główny kod wyświetla wyniki), to stanowi przykład doskonalszego stylu programowania. Funkcje powinny być uniwersalne i niezależne od specyfiki stron, na których są używane.
- Wyrażenie `return` zawsze kończy wykonanie kodu funkcji. Kod znajdujący się za wykonanym wyrażeniem `return` nie zostanie wykonany.
- W jednej funkcji może występować kilka wyrażeń `return` (na przykład w strukturze `switch` lub w wyrażeniu warunkowym), ale wykonane zostanie tylko jedno z nich. Funkcje często wykorzystują tę możliwość w następujący sposób:

```
function jakas_funkcja() {
  if (warunek) {
    return TRUE;
  } else {
    return FALSE;
  }
}
```

- Jeśli funkcja ma zwracać wiele wartości, to należy użyć tablicy i funkcji `list()`.

```
function calculate_total ($qty,
  $cost, $tax = 5) {
  $taxrate = $tax / 100;
  $total = ($qty * $cost) *
    ($taxrate + 1);
  return array ($total, $tax);
}
// Zwykły kod PHP
list ($total_cost, $taxrate) =
  calculate_total($_POST['quantity '],
$_POST['price'])
```

Zasięg zmiennej

Zasięg zmiennej to trudne, ale bardzo ważne pojęcie. Do każdej zmiennej w PHP przypisany jest jakiś zasięg, czyli obszar, w którym można się do niej odwoływać (a zatem także i do jej wartości). Ogólna zasada jest taka: zasięg zmiennej ograniczony jest do strony, w której ją umieszczono. Innymi słowy, po zdefiniowaniu zmiennej (nadaniu jej wartości) możesz się do niej odwoływać w dowolnym miejscu na tej stronie, podczas gdy inne strony nie mają do niej dostępu (chyba, że używasz specjalnych zmiennych).

Ponieważ pliki dołączane można traktować jako część oryginalnego (dołączającego je) skryptu, wszystkie zmienne zdefiniowane przed wyrażeniem `include` mogą być wykorzystywane w dołączanym pliku (co pokazałem już na przykładzie zmiennej `$page_title` i pliku *naglowek.html*). Ponadto zmienne zdefiniowane w dołączanym pliku są dostępne dla pliku dołączającego od miejsca wystąpienia wyrażenia `include`.

Wszystko to staje się jeszcze bardziej zagmatwane z chwilą, gdy zaczynasz definiować własne funkcje. Każda z nich ma swój własny zasięg, co oznacza, że zmienne wykorzystywane wewnątrz funkcji nie są dostępne poza nią i vice versa. Z tego względu zmienna występująca w ciele funkcji może mieć dokładnie taką samą nazwę jak zmienna leżąca poza nim. Są to jednak dwie zupełnie różne zmienne, które mogą mieć różne wartości. Koncepcja ta często sprawia trudności początkującym programistom.

Możesz też zmienić zasięg zmiennej występującej w funkcji, posługując się wyrażeniem `global`.

```
function nazwa_funkcji() {
    global $zmienna;
}
$zmienna = 20;
nazwa_funkcji(); // Wywołanie funkcji
```

W tym przykładzie `$zmienna` występująca wewnątrz funkcji i `$zmienna` poza nią to jedna i ta sama zmienna. Oznacza to, że jeżeli wartość zmiennej `$zmienna` zostanie zmieniona wewnątrz funkcji, to zmieni się wartość zewnętrznej zmiennej `$zmienna`.

Innym sposobem na ominięcie zasięgu zmiennej jest odwołanie się do zmiennych superglobalnych: `$_GET`, `$_POST`, `$_REQUEST` itd. Są one automatycznie dostępne we wszystkich Twoich funkcjach (dlatego właśnie nazywamy je *superglobalnymi*).

Aby wykorzystywać zmienne globalne:

1. Otwórz w edytorze tekstów plik *kalkulator.php* (patrz listing 3.10).
2. Zmień definicję funkcji na (listing 3.11):

```
function calculate_total($tax = 5) {
  global $total;
  $taxrate = $tax / 100;
  $total = ($_POST['quantity'] *
    $_POST['price']) *
    ($taxrate + 1);
  $total = number_format ($total, 2);
}
```

   Ponieważ i tak wykorzystywałem już zmienne superglobalne, mogę zmienić definicję funkcji tak, aby nie pobierała ona żadnych argumentów, tylko korzystała właśnie z superglobalnych (`$_POST['quantity']` i `$_POST['price']`). Na podobnej zasadzie mogę zrezygnować ze zwracania wartości `$total`, nadając jej zasięg globalny.

Listing 3.11. *Ponieważ $_POST jest zmienną superglobalną, do przechowywanych w niej wartości można się odwoływać we wszystkich funkcjach. Zmienna $total została zadeklarowana wewnątrz funkcji jako globalna*

Listing

```
1  <?php # Skrypt 3.11 - kalkulator.php (szósta wersja skryptów 3.5, 3.6, 3.8, 3.9 i 3.10)
2  $page_title = 'Kalkulator kosztów';
3  include ('./includes/naglowek.html');
4
5  /* Funkcja wyliczająca całkowity koszt
6  i wyświetlająca wynik. */
7  function calculate_total ($tax = 5) {
8    global $total;
9    $taxrate = $tax / 100; // Zamień 5% na .05.
10   $total = ($_POST['quantity'] * $_POST['price']) * ($taxrate + 1);
11   $total = number_format ($total, 2);
12 } // Koniec funkcji.
13
14 // Sprawdź czy formularz został wysłany.
15 if (isset($_POST['submitted'])) {
16
17   if (is_numeric($_POST['quantity']) && is_numeric($_POST['price'])) {
18
19     // Wyświetl nagłówek.
20     echo '<h1 id="mainhead">Całkowity koszt</h1>';
21
22     $total = NULL; // Zainicjalizuj $total.
23
24     if (is_numeric($_POST['tax'])) {
25       calculate_total ($_POST['tax']);
26     } else {
27         calculate_total ();
28     }
29
30     echo '<p>Kupujesz ' . $_POST['quantity'] . ' sztuk w cenie ' . number_format($_
       POST['price'], 2) . 'zł za egzemplarz. Całkowity koszt wyniesie ' . $total .  'zł.</p>';
31
32     // Wyświetl odstępy.
33     echo '<p><br /></p>';
34
35   } else { // Wprowadzono niepoprawne wartości.
36       echo '<h1 id="mainhead">Błąd!</h1>
37       <p class="error">Wprowadź poprawną liczbę egzemplarzy i cenę jednostkową.</p><p><br /></p>';
38   }
39
40 } // Koniec głównego IF.
41
42 // Koniec sekcji PHP i początek formularza HTML.
43 ?>
44 <h2>Kalkulator kosztów</h2>
45 <form action="kalkulator.php" method="post">
46   <p>Liczba egzemplarzy: <input type="text"  name="quantity" size="5"   maxlength="10" value="<?php
     if (isset($_POST['quantity'])) echo  $_POST['quantity']; ?>" /></p>
47   <p>Cena: <input type="text"  name="price" size="5"  maxlength="10" value="<?php if
     (isset($_POST['price'])) echo  $_POST['price']; ?>" /> </p>
48   <p>Podatek:</b> <input type="text"  name="tax" size="5" maxlength="10"  value="<?php if
     (isset($_POST  ['tax'])) echo $_POST['tax'];?>" />(opcjonalny)</p>
```

3. Zainicjuj zmienną $total na zewnątrz funkcji.

```
$total = NULL;
```

Dobry styl programowania wymaga, aby, zanim funkcja użyje zmiennej jako globalnej, zainicjować ją wartością NULL.

4. Zmień sposób wywołania funkcji na:

```
if (is_numeric($_POST['tax'])) {
  calculate_total($_POST['tax']);
} else {
  calculate_total();
}
```

5. Zmień wyrażenie echo() na:

```
echo '<p>Kupujesz ' . {$_POST
  ['quantity']  . ' sztuk w cenie ' .
  number_format($_POST['price'], 2) .
  ' zł za egzemplarz. Po uwzględnieniu
  podatku daje to ' . $total '
  zł.</p>';
```

Zamiast odwoływać się do zmiennej $total_cost (której wartość była poprzednio zwracana przez funkcję), w tym przykładzie wyświetlana jest wartość zmiennej globalnej $total.

6. Zapisz plik, wgraj go na serwer i przetestuj w przeglądarce internetowej (rysunek 3.18).

Listing 3.11. *Ponieważ $_POST jest zmienną superglobalną, do przechowywanych w niej wartości można się odwoływać we wszystkich funkcjach. Zmienna $total została zadeklarowana wewnątrz funkcji jako globalna — ciąg dalszy*

Listing

```
49    <p><input type="submit" name="submit"
      value="Oblicz!" /></p>
50    <input type="hidden" name="submitted"
      value="TRUE" />
51  </form>
52  <?php
53  include ('./includes/stopka.html');
54  ?>
```

Rysunek 3.18. *Po raz kolejny ta sama strona została wygenerowana przy użyciu innej wersji funkcji*

Wskazówki

- Inny aspekt zasięgu zmiennej związany jest z miejscem definicji funkcji. Pamiętajmy, że kod funkcji zostaje wykonany w miejscu jej wywołania, a nie w miejscu definicji. Wystarczy więc, że zmienną globalną zdefiniujemy przed wywołaniem funkcji, która jej używa.
- Zmienna *statyczna* to taka, która należy do funkcji, jest inicjalizowana przy pierwszym wywołaniu danej funkcji i której wartość jest pamiętana między kolejnymi wywołaniami. Przy pierwszym wywołaniu poniższej funkcji zostanie wyświetlona liczba `1`, przy drugim `2` itd.

```
function licznik () {
    static $var = 1;
    echo $var++;
}
```

- Istnieje jeszcze jeden sposób na „poszerzenie" zasięgu zmiennej. Zamiast przekazywać zmienne do funkcji przez wartość, należy przekazać je przez referencję. Dzięki temu wszelkie zmiany ich wartości będą też widziane „na zewnątrz" funkcji. Więcej informacji na temat przekazywania zmiennych przez referencję znajdziesz w podręczniku PHP.
- W PHP możesz też korzystać z tablicy `$GLOBALS` przechowującej wszystkie zmienne globalne skryptu.
- Zmienne `$_SESSION` i `$_COOKIE` mogą być dostępne dla wielu stron. Więcej na ten temat w rozdziale 9., „Sesje i „ciasteczka"".
- Choć za każdym razem możesz przepisać swe funkcje w taki sposób, aby wykorzystywać w nich wyrażenie `global` i zmienne superglobalne, nie zawsze ma to sens. Poprzednia wersja funkcji była lepsza, ponieważ akceptowała trzy wartości i zwracała jedną bez względu na to, jakie były nazwy pól wejściowych formularza (odpowiadające zmiennym `$_POST`).

Funkcje daty i czasu

Omówię teraz kilka funkcji związanych z datą i czasem. Najważniejsza z nich to `date()`, zwracająca odpowiednio sformatowany łańcuch znaków określający datę i godzinę.

```
date (format, [znacznik_czasu]);
```

znacznik_czasu to opcjonalny argument pobierający datę wyrażoną przez liczbę sekund, jaka upłynęła od początku Epoki Uniksa (czyli od północy 1 stycznia 1970 r.). Jego zastosowanie pozwala nam uzyskać informacje (np. dzień tygodnia) dla dowolnej daty. W przeciwnym razie PHP zwróci bieżącą datę ustawioną na serwerze.

Funkcja `date()` obsługuje mnóstwo parametrów formatujących (tabela 3.1), które można łączyć ze zwykłym tekstem. Na przykład:

```
echo date('F j, Y'); // January 21, 2003
echo date('H:i'); // 23:14
echo date('Dziś jest D'); // Dziś jest Mon
```

Za pomocą funkcji `mktime()` możesz obliczyć znacznik czasu dla wskazanej daty.

```
int mktime (godzina, minuta, sekunda,
  miesiac, dzien, rok);
```

Z kolei funkcja `getdate()` zwraca tablicę wartości (tabela 3.2) opisujących bieżącą datę i czas. Na przykład:

```
$dates = getdate();
echo $dates['month']; // January
```

Również ta funkcja posiada opcjonalny argument w postaci znacznika czasu. Jeśli nie zostanie on użyty, funkcja `getdate()` zwróci bieżący czas i datę.

Aby przećwiczyć posługiwanie się tymi funkcjami, przepiszemy skrypt *kalendarz.php* w taki sposób, aby wyświetlał wstępnie wybraną bieżącą datę.

Tabela 3.1. *Każdy z tych parametrów wpływa na format, w jakim funkcja date() zwraca wyniki*

Znaczniki formatujące dla dat		
Znak	**Znaczenie**	**Przykład**
Y	rok wyrażony 4 cyframi	2003
y	rok wyrażony 2 cyframi	03
n	miesiąc wyrażony 1 lub 2 cyframi	2
m	miesiąc wyrażony 2 cyframi	02
F	miesiąc	February
M	miesiąc wyrażony 3 literami	Feb
j	dzień miesiąca wyrażony 1 lub 2 cyframi	8
d	dzień miesiąca wyrażony 2 cyframi	08
l (małe L)	dzień tygodnia	Monday
D	dzień tygodnia wyrażony 3 literami	Mon
g	godzina w formacie 12-godzinnym, wyrażona 1 lub 2 cyframi	6
G	godzina w formacie 24-godzinnym, wyrażona 1 lub 2 cyframi	18
h	godzina w formacie 12-godzinnym, wyrażona 2 cyframi	06
H	godzina w formacie 24-godzinnym, wyrażona 2 cyframi	18
i	minuty	45
s	sekundy	18
a	am lub pm	am
A	AM lub PM	PM

Tabela 3.2. *Funkcja getdate() zwraca tablicę asocjacyjną*

Tablica getdate()		
Klucz	**Wartość**	**Przykład**
year	rok	2005
mon	miesiąc	12
month	nazwa miesiąca	December
mday	dzień miesiąca	25
weekday	dzień tygodnia	Tuesday
hours	godziny	11
minutes	minuty	56
seconds	sekundy	47

Listing 3.12. *Skrypt wykorzystuje teraz funkcje date() i getdate()*

Listing

```
<?php # Skrypt 3.12 - kalendarz.php
(druga wersja skryptu 3.7)
$page_title = 'Kalendarz';
include ('./includes/naglowek.html');

// Funkcja tworząca trzy menu
rozwijalne wyboru miesięcy, dni i lat.
function make_calendar_pulldowns($m =
NULL, $d = NULL, $y = NULL) {

  // Utwórz tablicę miesięcy.
  $months = array (1 => 'Styczeń',
  'Luty', 'Marzec', 'Kwiecień', 'Maj',
  'Czerwiec', 'Lipiec', 'Sierpień',
  'Wrzesień', 'Październik',
  'Listopad', 'Grudzień');

  // Utwórz menu miesięcy.
  echo '<select name="month">';
  foreach ($months as $key => $value) {
    echo "<option value=\"$key\"";
    if ($key == $m) { // Wstępny wybór.
      echo ' selected="selected"';
    }
    echo ">$value</option>\n";
  }
  echo '</select>';

  // Utwórz menu dni.
  echo '<select name="day">';
  for ($day = 1; $day <= 31; $day++) {
      echo "<option value=\"$day\"";
      if ($day == $d) { // Wstępny
      wybór.
        echo ' selected="selected"';
      }
      echo ">$day</option>\n";
  }
  echo '</select>';

  // Utwórz menu lat.
  echo '<select name="year">';
  for ($year = 2005; $year <= 2015;
  $year++) {
```

Aby wykorzystać funkcje operujące na dacie:

1. Utwórz w edytorze tekstów skrypt *kalendarz.php* (listing 3.7).
2. Zmień definicję funkcji na następującą (listing 3.12):

```
function make_calendar_pulldowns
  ($m = NULL, $d = NULL, $y = NULL) {
```

   Definiowana funkcja pobiera do trzech argumentów reprezentujących miesiąc, dzień i rok. Ich zadaniem będzie wybór wstępnych pozycji w menu rozwijalnym w podobny sposób, jak w formularzach zapamiętujących dane.

3. Zmień pętlę `foreach` tak, aby wstępnie wybierała miesiąc.

```
foreach ($months as $key => $value) {
  echo "<option value=\"$key\"";
  if ($key == $m) {
    echo ' selected="selected"';
  }
  echo ">$value</option>\n";
}
```

   Proces tworzenia menu rozwijalnego praktycznie się nie zmienił, poza dodaniem wyrażenia warunkowego. Jeśli funkcja otrzyma wartość `$m`, to w formularzu powinien zostać wybrany odpowiedni miesiąc. W tym celu wartość `$m` porównywana jest z kluczami kolejnych elementów tablicy. Jeśli jest równa, to do odpowiedniego znacznika `option` dodawany jest atrybut `selected="selected"`.

4. W podobny sposób zmodyfikuj pętle `for` dla menu wyboru dnia.

```
for ($day = 1; $day <= 31; $day++) {
  echo "<option value=\"$day\"";
  if ($day == $d) {
    echo ' selected="selected"';
  }
  echo ">$day</option>\n";
}
```

Kod ten działa tak samo jak w przypadku miesięcy. Inna jest tylko pętla. Jeśli `$d` jest równe `$day`, to do odpowiedniego znacznika `option` dodawany jest atrybut `selected="selected"`.

5. W taki sam sposób zmodyfikuj pętlę lat.

```
for ($year = 2005; $year <= 2015;
  $year++) {
  echo "<option value=\"$year\"";
  if ($year == $y) {
    echo ' selected="selected"';
  }
  echo ">$year</option>\n";
}
```

6. Zmień wywołanie funkcji w taki sposób, aby została wywołana dla bieżącej daty.

```
$dates = getdate();
make_calendar_pulldowns
($dates['mon'], $dates['mday'],
  $dates['year']);
```

Przed przekazaniem do funkcji parametrów określających bieżący dzień, miesiąc i rok wywołam funkcję `getdate()`, która zwróci tablicę z wartościami odpowiadającymi dzisiejszej dacie. Następnie wywołam funkcję, wykorzystując odpowiednie elementy tablicy.

7. Wyświetl bieżącą datę i godzinę.

```
echo '<p>Dziś jest ', date ('l'),
  '. Jest godzina ', date ('g:i a'),
  '.</p>';
```

Ostatnie wyrażenie spowoduje przesłanie do przeglądarki następujących danych:

```
<p>Dziś jest Tuesday.
  Jest godzina 11:14pm.</p>
```

Zostaną przy tym uwzględnione parametry formatujące przekazane do funkcji `date()`.

Listing 3.12. *Skrypt wykorzystuje teraz funkcje date() i getdate() — ciąg dalszy*

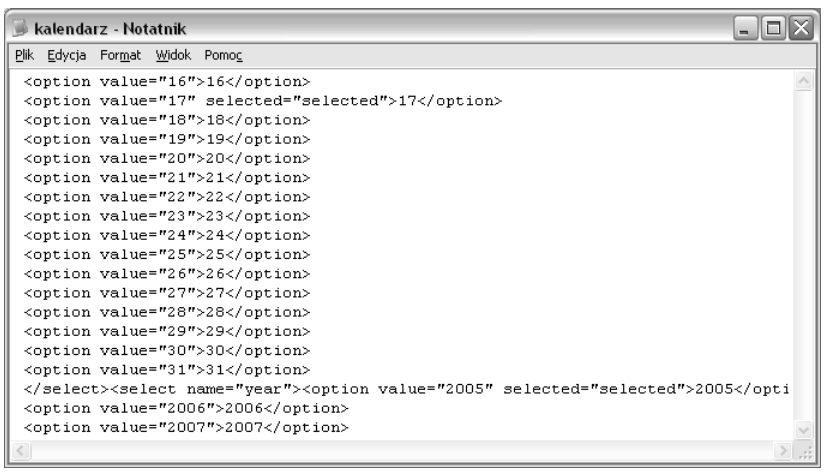
```
Listing
36      echo "<option value=\"$year\"";
37      if ($year == $y) { // Wstępny wybór.
38        echo ' selected="selected"';
39      }
40      echo ">$year</option>\n";
41    }
42    echo '</select>';
43
44  } // Koniec definicji funkcji.
45
46  // Utwórz znaczniki formularza
47  echo '<h1 id="mainhead">Wybierz
    datę:</h1>
48  <p><br /></p><form
    action="kalendarz.php" method="post">';
49
50  // Pobierz informację o dniu dzisiejszym
    i wywołaj funkcję.
51  $dates = getdate();
52  make_calendar_pulldowns ($dates['mon'],
    $dates['mday'], $dates['year']);
53
54  echo '</form><p><br /></p>'; // Koniec
    formularza.
55
56  // Wyświetl bieżącą datę i czas.
57  echo '<p>Dziś jest ' . date ('l') . '.
    Jest godzina ' . date ('g:i a') .
    '.</p>';
58
59  include ('./includes/stopka.html');
60  ?>
```

Rysunek 3.19. *Funkcje date() i getdate() pomogły mi wprowadzić nieco życia na tę stronę*

8. Zapisz plik pod nazwą *kalendarz.php*, wgraj go na serwer i przetestuj w przeglądarce internetowej (rysunek 3.19).
9. Możesz podejrzeć kod źródłowy strony, aby zobaczyć, w jaki sposób menu zapamiętują dane (rysunek 3.20).

Wskazówki

- Jak zobaczysz w rozdziale 4., „Wprowadzenie do SQL i MySQL”, MySQL ma własne funkcje operujące na datach.
- Funkcje daty PHP odzwierciedlają czas serwera PHP. Jeżeli chcesz sprawdzić bieżącą datę i czas ustawione na komputerze klienckim, musisz posłużyć się JavaScriptem.
- Funkcja `checkdate()` pobiera trzy parametry — miesiąc, dzień i rok i sprawdza, czy określają one poprawną datę.



Rysunek 3.20. *Kod źródłowy strony ujawnia sposób zapamiętywania wybranych pozycji menu za pomocą atrybutu selected="selected"*

Wysyłanie poczty elektronicznej

Jedną z moich ulubionych cech PHP jest łatwość, z jaką można w tym języku wysyłać e-maile. Na prawidłowo skonfigurowanym serwerze sprowadza się to do wywołania funkcji `mail()`.

```
mail ($do, $temat, $tresc);
```

Zmienna `$do` powinna zawierać adres e-mail lub listę adresów oddzielanych przecinkami. Zmienna `$temat` przechowuje temat listu, a w zmiennej `$tresc` umieszcza się właściwą wiadomość. Może ona zawierać ujęte w cudzysłowy znaki nowego wiersza (`\n`), dzięki czemu treść wiadomości będzie wyświetlana na ekranie w kilku wierszach.

Funkcja `mail()` ma też czwarty, opcjonalny parametr, w którym mogą znaleźć się dodatkowe nagłówki wiadomości (*From*, *Reply-To*, *Cc*, *Bcc*, etc.). Na przykład:

```
mail ('phpmysql@dmcinsights.com',
  'Pytanie dotyczące listingu 3.13',
  $tresc, 'From: osoba@adres.com');
```

Jeżeli chcesz użyć kilku różnych nagłówków, oddziel je od siebie znakami `\r\n`:

```
$naglowki = 'From: Jan@Kowalski.pl\r\n';
$naglowki .= 'Cc: Janina@Kowalska.pl,
  Janina@Kowalska\r\n';
mail ('phpmysql@dmcinsights.com',
  'Pytanie dotyczące listingu 3.13',
  $tresc, $naglowki);
```

W ostatnim przykładzie zamieszczonym w tym rozdziale stworzę prosty formularz rejestracji. Jeśli rejestracja zakończy się pomyślnie, to wyśle on wiadomość pocztą elektroniczną. W przykładzie tym zademonstruję również prosty sposób obsługi i raportowania wielu błędów związanych z wysłaniem formularza.

Aby wysłać e-mail:

1. Rozpocznij w edytorze tekstów nowy skrypt PHP (patrz listing 3.13).

   ```
   <?php # Script 3.13 - register.php
   $page_title = 'Rejestracja';
   include ('./includes/naglowek.html');
   ```

2. Utwórz wyrażenie warunkowe sprawdzające, czy formularz został wysłany oraz zmienną przechowującą błędy rejestracji.

   ```
   if (isset($_POST['submitted'])) {
     $errors = array();
   ```

   Zmienna `$errors` będzie przechowywać wszystkie błędy związane z weryfikacją danych formularza. Zostaje zainicjowana jako tablica, co nie jest konieczne, ale stanowi przykład dobrego stylu programowania.

3. Sprawdź, czy użytkownik wprowadził swoje nazwisko i adres poczty elektronicznej.

   ```
   if (empty($_POST['name'])) {
     $errors[] = 'Zapomniałeś wprowadzić
     nazwisko.';
     }
   if (empty($_POST['email'])) {
     $errors[] = 'Zapomniałeś wprowadzić adres
   poczty elektroniczej.';
   }
   ```

   Weryfikacja tych dwóch pól odbywa się w najprostszy sposób poprzez sprawdzenie, czy nie są puste. Jeśli któreś z nich jest puste, to do tablicy `$errors` zostaje dodany odpowiedni komunikat.

   W rozdziale 10., „Zabezpieczenia”, dowiesz się, jak za pomocą wyrażeń regularnych można weryfikować adresy e-mail wprowadzane przez użytkowników.

Listing 3.13. *Funkcja mail() jest zaskakująco prosta w użyciu*

Listing

```
1  <?php # Skrypt 3.13 - register.php
2  $page_title = 'Zarejestruj się';
3  include ('./includes/naglowek.html');
4
5  // Sprawdź czy formularz został wysłany.
6  if (isset($_POST['submitted'])) {
7
8    $errors = array(); // Zainicjalizuj tablicę błędów.
9
10   // Zweryfikuj nazwisko.
11   if (empty($_POST['name'])) {
12   $errors[] = 'Zapomniałeś wprowadzić nazwisko.';
13 }
14
15 // Zweryfikuj adres poczty elektronicznej.
16 if (empty($_POST['email'])) {
17   $errors[] = 'Zapomniałeś wprowadzić adres poczty elektronicznej.';
18 }
19
20 // Zweryfikuj hasło.
21 if (!empty($_POST['password1'])) {
22   if ($_POST['password1'] != $_POST['password2']) {
23     $errors[] = 'Hasła nie są takie same.';
24   }
25 } else {
26     $errors[] = 'Zapomniałeś wprowadzić hasła.';
27 }
28
29 if (empty($errors)) { // Jeśli nie ma błędów.
30
31   // Zarejestruj użytkownika.
32
33   // Wyślij wiadomość pocztą elektroniczną.
34   $body = "Dziękujemy za zarejestrowanie się na naszej stronie!\nTwoje hasło to
     '{$_POST['password1']}'.\n\nZ  poważaniem,\nMy";
35   mail ($_POST['email'], 'Dziękujemy za zarejestrowanie się!', $body,
     'From:admin@strona.com');
36
37   echo '<h1 id="mainhead">Dziękujemy!</h1>
38   <p>Zostałeś zarejestrowany. Potwierdzenie zostało wysłane pocztą elektroniczną.</p><p><br
     /></p>';
39
40 } else { // Raportuj błędy.
41
42     echo '<h1 id="mainhead">Błąd!</h1>
43     <p class="error">Wystąpiły następujące błędy:<br />';
44     foreach ($errors as $msg) { // Wyświetl każdy błąd.
45       echo " - $msg<br />\n";
46     }
47     echo '</p><p>Wypełnij formularz jeszcze raz.</p><p><br /></p>';
48
49   } // Koniec if (empty($errors))
50
51 } else { // Wyświetl formularz.
```

4. Zweryfikuj hasło.

```
if (!empty($_POST['password1'])) {
  if ($_POST['password1'] != $_POST
    ['password2']) {
    $errors[] = 'Hasła nie są takie same.';
  }
} else {
  $errors[] = 'Zapomniałeś wprowadzić
  hasła.';
}
```

Weryfikacja hasła odbywa się w dwóch etapach. Najpierw sprawdzam, czy nie jest ono puste, a następnie, czy hasło wprowadzone po raz drugi jest takie samo. Jeśli któryś z tych warunków nie jest spełniony, to do tablicy `$errors` wstawiany jest kolejny komunikat o błędzie.

5. Sprawdź, czy podczas weryfikacji danych wystąpiły błędy.

```
if (empty($errors)) {
```

Jeśli nie, to tablica błędów jest pusta i można wysłać potwierdzenie rejestracji jako wiadomość poczty elektronicznej.

6. Wyślij wiadomość poczty elektronicznej.

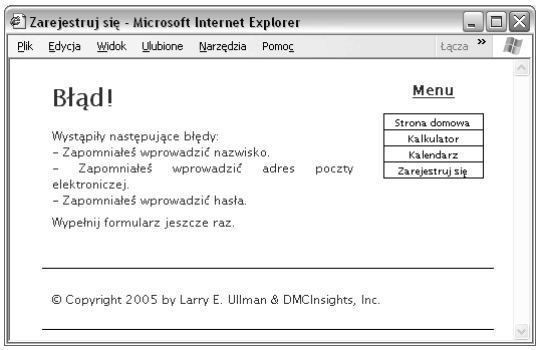
```
$body = "Dziękujemy za zarejestrowanie się
  na naszej stronie!\nTwoje hasło to
  '{$_POST['password1']}'.\n\nZ
  poważaniem,\nMy";
mail ($_POST['email'], 'Dziękujemy
  za zarejestrowanie się!', $body,
  'From:admin@strona.com');
```

Przed wysłaniem wiadomości składam jej treść i przypisuję ją do zmiennej `$body`. Dopiero wtedy wywołuję funkcję `mail()`.

Listing 3.13. *Funkcja mail() jest zaskakująco prosta w użyciu — ciąg dalszy*

Listing

```
52 ?>
53 <h2>Rejestracja</h2>
54 <form action="rejestracja.php"
   method="post">
55   <p>Nazwisko: <input type="text"
     name="name" size="20" maxlength="40"
     /></p>
56   <p>E-mail: <input type="text"
     name="email" size="20"
     maxlength="40" /> </p>
57   <p>Hasło: <input type="password"
     name="password1" size="10"
     maxlength="20" /></p>
58   <p>Potwierdź hasło: <input
     type="password" name="password2"
     size="10" maxlength="20" /></p>
59   <p><input type="submit"
     name="submit" value="Zarejestruj
     się" /></p>
60   <input type="hidden"
     name="submitted" value="TRUE" />
61 </form>
62 <?php
63 } // Koniec głównego wyrażenia
   warunkowego.
64 include ('./includes/stopka.html');
65 ?>
```



Rysunek 3.21. *Jeśli formularz nie będzie wypełniony w całości, wyświetlone zostaną komunikaty o błędzie*

7. Wyświetl komunikat.

```
echo '<h1 id="mainhead">Dziękujemy!</h1>
<p>Zostałeś zarejestrowany.
Potwierdzenie zostało wysłane pocztą
elektroniczną.
</p><p><br /></p>';
```

Na razie skrypt nie rejestruje jeszcze użytkownika, ale funkcjonalność tę dodamy wkrótce.

8. Uzupełnij wewnętrzne wyrażenie warunkowe raportem błędów.

```
} else {
  echo '<h1 id="mainhead">Blad!</h1>
  <p class="error">Wystąpiły
  następujące błędy:<br />';
  foreach ($errors as $msg) {
    echo " - $msg<br />\n";
  }
  echo '</p><p>Wypełnij formularz
    jeszcze raz.</p><p><br /></p>';
}
```

Ponieważ zmienna `$errors` jest tablicą, mogę łatwo wyświetlić wszystkie komunikaty o błędach, używając pętli (rysunek 3.21).

9. Zakończ główne wyrażenie warunkowe i zamknij znaczniki PHP.

```
} else {
?>
```

W ten sposób zostało zamknięte wyrażenie warunkowe sprawdzające czy formularz został wysłany. Teraz, na zewnątrz znaczników PHP, stworzę formularz.

10. Utwórz formularz HTML.

```
<h2>Rejestracja</h2>
<form action="register.php"
  method="post">
  <p>Nazwisko: <input type="text"
    name="name" size="20"
    maxlength="40" /></p>
  <p>Adres poczty elektronicznej: <input
    type="text" name="email"
    size="20" maxlength="40" /> </p>
  <p>Hasło: <input type="password"
    name="password1" size="10"
    maxlength="20" /></p>
  <p>Potwierdź hasło: <input
    type="password" name="password2"
    size="10" maxlength="20" /></p>
  <p><input type="submit"
    name="submit" value=
    "Zarejestruj się" /></p>
  <input type="hidden" name="submitted"
    value="TRUE" />
</form>
```

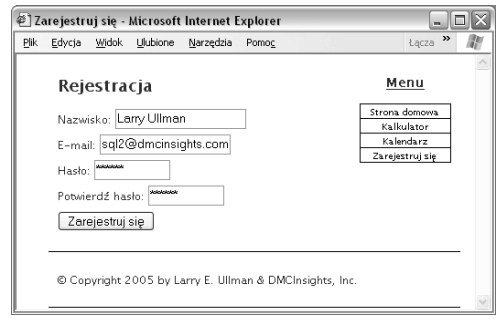
Brak tu specjalnych nowości, może oprócz konieczności dwukrotnego wprowadzenia hasła. Jest to konieczne, ponieważ znaki hasła nie są widoczne podczas ich wprowadzania (rysunek 3.22). Gdyby użytkownik wprowadzał hasło tylko jeden raz i pomylił się, nie wiedziałby jakie hasło w rzeczywistości wprowadził.

11. Zakończ stronę PHP.

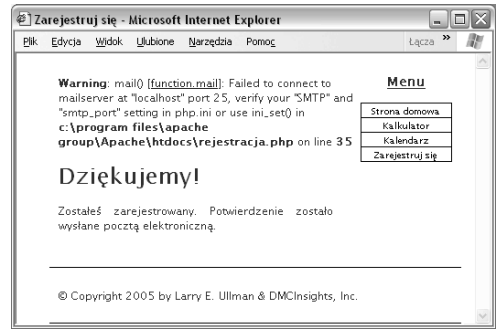
```
<?php
}
include ('./includes/footer.html');
?>
```

Nawias klamrowy zamyka główne wyrażenie warunkowe (które sprawdza, czy formularz został wysłany).

12. Zapisz plik pod nazwą *rejestracja.php*, wgraj go na serwer i przetestuj w przeglądarce internetowej (rysunek 3.23).



Rysunek 3.22. *Formularz rejestracji…*



Rysunek 3.23. *... po pomyślnym wypełnieniu*

Rysunek 3.24. *E-mail wysłany do mnie przez funkcję mail()*

Jeśli wystąpi dziwny komunikat o błędzie (*From header missing* lub *No SMTP server*), lub nie dostaniesz wiadomości poczty elektronicznej z potwierdzeniem rejestracji, zapoznaj się z treścią ramki „Funkcja mail() a konfiguracja poczty elektronicznej"

13. Sprawdź, czy dostałeś potwierdzenie pocztą elektroniczną (rysunek 3.24).

Wskazówki

- W niektórych systemach, głównie pochodnych systemu Unix, zamiast sekwencji `\r\n` powinieneś używać samego `\n`.
- Funkcja `mail()` zwraca wartość `0` lub `1`, informując tym samym, że została ona prawidłowo wywołana. Nie oznacza to jednak jeszcze, że udało się wysłać e-maile ani że zostały one odebrane przez adresata.
- Gdy wykorzystujesz funkcję `mail()` w sposób pokazany w tym rozdziale, nie możesz wysyłać wiadomości z załącznikami ani stosować formatowania HTML. Jeżeli zależy Ci na wysyłaniu załączników, musisz wykorzystać odpowiednią klasę MIME. Poszukaj w rozdziale 11. informacji na temat PEAR.

Funkcja mail() a konfiguracja poczty elektronicznej

Funkcja `mail()` używana w skryptach PHP nie wysyła sama wiadomości poczty elektronicznej, lecz przekazuje to zadanie serwerowi poczty elektronicznej. Oznacza to, że Twój komputer musi mieć działający serwer poczty elektronicznej, aby funkcja `mail()` działała poprawnie.

Jeśli Twój komputer pracuje pod kontrolą jednego z wariantów systemu Unix lub gdy używasz zewnętrznego, komercyjnego serwera WWW, to korzystanie z funkcji `mail()` nie powinno przysparzać kłopotów. Natomiast jeśli wykonujesz skrypty PHP na własnym komputerze, to prawdopodobnie będziesz musiał go dodatkowo skonfigurować

Jeśli Twój komputer pracuje pod kontrolą systemu Windows i masz konto u dostawcy usług internetowych udostępniającego serwer SMTP (na przykład *smtp.comcast.net*), to powinieneś skonfigurować go w pliku *php.ini*. Jednak rozwiązanie takie będzie działać tylko pod warunkiem, że serwer ten nie wymaga uwierzytelniania za pomocą nazwy i hasła. W takim przypadku będziesz musiał zainstalować własny serwer SMTP. Dostępnych jest wiele różnych implementacji serwera SMTP (wyszukaj w internecie hasło *free windows smtp server*), a przekonasz się, że ich instalacja nie jest trudna.