W rozdziale 10. utworzyliśmy stronę WWW z wykorzystaniem szablonów. Na stronie głównej tego prostego serwisu umieściliśmy aktualności pobierane z bazy danych. W tym rozdziale, wykorzystując umiejętności zdobyte podczas kursu, napiszemy prosty system zarządzania treścią na potrzeby strony firmowej z rozdziału 10. Programy tego typu (choć zwykle o wiele bardziej zaawansowane niż nasz) określane są jako CMS (ang. Content Management System — system zarządzania zawartością). W Internecie można znaleźć wiele gotowych CMS-ów, często wchodzących w skład aplikacji WWW do tworzenia serwisów, portali, forów itp.

Nasz CMS będzie prosty, ponieważ ma on służyć celom czysto szkoleniowym — chodzi o użycie w praktyce metod dostępu do danych, przekazywania różnych wartości pomiędzy stronami i śledzenia sesji.

Na początku przedstawiono projekt serwisu, następnie opisano budowę poszczególnych stron i przetestowano działanie gotowego serwisu.

Projekt systemu

System zarządzania wiadomościami (tak go będziemy roboczo nazywać, ponieważ nazwa CMS jest jednak dla niego troszkę za poważna) będzie umożliwiał wykonanie operacji dodawania wiadomości, edycji istniejącej wiadomości i usuwania wiadomości. Wszystkie te operacje będą dostępne oczywiście dopiero po zalogowaniu się użytkownika.

Po zalogowaniu się użytkownik przeniesiony zostanie na stronę główną systemu, gdzie będzie mógł się przenieść na podstrony związane z operacjami dodawania, edycji i usuwania informacji. Ze strony głównej oraz poszczególnych podstron użytkownik będzie mógł się wylogować. W przypadku poszczególnych podstron możliwy będzie powrót do strony „macierzystej” (nadrzędnej) — w przypadku tego typu serwisów starajmy się raczej nie używać do powracania do poprzedniej strony przycisków dostępnych w przeglądarce, ponieważ system może się zachowywać nieprzewidywalnie.

Po przejściu do podstrony dodawania wiadomości użytkownik będzie miał do dyspozycji formularz, gdzie będzie mógł wprowadzić tytuł wiadomości oraz jej treść (posługujemy się bazą danych z rozdziału 7. i 10.), data będzie automatycznie ustawiana na aktualną datę serwera. Po zatwierdzeniu przez użytkownika wprowadzonych danych przyciskiem o nazwie np. Dodaj dane zostaną przesłane do skryptu, który doda je do bazy i poinformuje o tym użytkownika, przekierowując go do strony głównej systemu.

Wybór edycji spowoduje przeniesienie użytkownika do podstrony edycji, na której wypisane będą wszystkie wiadomości w kolejności od najnowszej do najstarszej. Poniżej każdej wiadomości umieszczone zostanie łącze (o nazwie Edytuj), którego kliknięcie przeniesie użytkownika do formularza edycji wybranej wiadomości. Od tego miejsca wszystko przypominać będzie dodawanie wiadomości — na końcu użytkownik zostanie poinformowany o wykonaniu operacji i przekierowany na podstronę wyboru wiadomości do edycji.

Po wejściu na podstronę usuwania wiadomości użytkownik zobaczy listę wiadomości (od najmłodszej do najstarszej). Pod każdą wiadomością będzie umieszczone łącze Usuń, którego kliknięcie spowoduje usunięcie danej wiadomości.

Ponadto nieuprawniony dostęp do stron (poza stroną logowania) będzie powodował automatyczne przeniesienie na stronę logowania. Nieuprawniony dostęp to próba wejścia do systemu bez uwierzytelnienia.

Tak z grubsza przedstawiają się założenia dotyczące naszego systemu. Zacznijmy więc tworzenie aplikacji.

Uzupełnienie bazy danych

Wykorzystamy oczywiście przygotowaną wcześniej bazę danych (rozdział 7.), w której utworzono już tabelę aktualnosci. Na potrzeby naszego systemu musimy dodać jeszcze jedną tabelę o nazwie uzytkownicy, która będzie przechowywać dane użytkowników (nazwę i hasło). Tabelę tę utworzyć można np. za pomocą polecenia z listingu 13.1.

Listing 13.1. Polecenie tworzące nową tabelę w bazie danych

```
CREATE SEQUENCE uzytkownicy_sq

  INCREMENT 1

  MINVALUE 1

  MAXVALUE 65535

  START 1

  CACHE 1;

ALTER TABLE uzytkownicy_sq OWNER TO postgres;



CREATE TABLE uzytkownicy

(

  uzytkownicy_id int4 NOT NULL DEFAULT  nextval('uzytkownicy_sq'::regclass),

  uzytkownicy_login varchar(12) NOT NULL,

  uzytkownicy_passwd varchar(32) NOT NULL,

  CONSTRAINT uzytkownicy_pkey_constraint  PRIMARY KEY (uzytkownicy_id)

)

WITHOUT OIDS;

ALTER TABLE uzytkownicy OWNER TO postgres;
```

Warto też, korzystając z programu pgAdmin, dodać do nowej tabeli wpis dla użytkownika rafal (będziemy go wykorzystywać do testowania systemu). Użytkownik ten powinien mieć hasło rafal123, które po „przepuszczeniu” przez funkcję md5() da nam ciąg 2cf2181059d7c9b2c94ec589d1d0356b i ten właśnie ciąg będziemy przechowywać w bazie jako hasło. Dodanie takiego użytkownika można wykonać za pomocą instrukcji SQL:

```
INSERT INTO uzytkownicy (uzytkownicy_login,  uzytkownicy_passwd) VALUES ('rafal',
'2cf2181059d7c9b2c94ec589d1d0356b');
```

Kod

Kolejne listingi (od 13.2 do 13.11) przedstawiają strony i skrypty wchodzące w skład aplikacji. Każdy z listingów opatrzony jest krótkim komentarzem — warto przyjrzeć się zarówno kodowi skryptów, jak i komentarzom. Wszystkie strony i skrypty z tych listingów powinny znajdować się w jednym katalogu — w przypadku aplikacji testowej będzie to katalog o nazwie Serwis (oczywiście nazwa katalogu może być dowolna). Dla uproszczenia strony kodowane są w języku HTML.

Listing 13.2. Strona logowania (index.html)

```
<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD  HTML 4.01//EN&quot;

  &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>

<html lang=&quot;pl&quot;>

<head>

 <meta  http-equiv=&quot;Content-Type&quot; content=&quot;text/html;  charset=utf-8&quot;>

 <title>Logowanie</title>

</head>

<body>

 <form action=&quot;main.php&quot;  method=&quot;POST&quot;>

  <div>

  Użytkownik:<br />

  <input name=&quot;user&quot;  /><br />

  Hasło:<br />

  <input name=&quot;passwd&quot;  type=&quot;password&quot; /><br />

  <input type=&quot;submit&quot;  value=&quot;Loguj&quot; name=&quot;login&quot; />

  </div>

 </form>

</body>

</html>
```

Strona logowania (listing 13.2) to zwykła strona w języku HTML, zawierająca formularz logowania. Dane logowania przekazane zostaną za pomocą metody POST protokołu HTTP skryptowi main.php (listing 13.3).

Listing 13.3. Strona główna systemu (main.php)

```
<?php

 session_start();

?>

<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD  HTML 4.01//EN&quot;

  &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>
```

```
<html lang=&quot;pl&quot;>

<head>

  <meta  http-equiv=&quot;Content-Type&quot; content=&quot;text/html;  charset=utf-8&quot;>

  <title>System zarządzania  aktualnościami</title>

</head>

<body>

  <h2>System zarządzania aktualnościami</h2>

  <div>

  <hr />

<?php

  if (isset($_POST['login'])) {

    $conn = pg_connect(&quot;dbname=SerwisWWW  user=postgres password=test&quot;);

    $user = addslashes(trim($_POST['user']));

    $passwd =  addslashes(trim($_POST['passwd']));

    $passwd = md5($passwd);

    $sql = &quot;SELECT * FROM uzytkownicy  WHERE uzytkownicy_login='$user' AND
uzytkownicy_passwd='$passwd'&quot;;

    $res = pg_query($conn, $sql);

    $ile = pg_num_rows($res);

    if ($ile == 1) {

      $_SESSION['valid_user'] = true;

      $tab = pg_fetch_assoc($res);

      $_SESSION['user_name'] =  $tab['uzytkownicy_login'];

    }

    else

      $_SESSION['valid_user'] = false;

    pg_close($conn);

  }
```

```
 if (!$_SESSION['valid_user']) {
  unset($_SESSION['valid_user']);
  session_destroy();
  //header(&amp;quot;Location:  index.html&amp;quot;);
 }
 else {
?>
 <p>
  Użytkownik: <b><?=  $_SESSION['user_name']; ?></b> [<a  href=&amp;quot;logout.php&amp;quot;>Wyloguj</a>]
 </p>
 <hr />
 <div>
  Wybierz czynność:<br />
  <ul>
   <li><a href=&amp;quot;add.php&amp;quot;>Dodaj nową  wiadomość</a></li>
   <li><a href=&amp;quot;edit_lst.php&amp;quot;>Edytuj  wiadomość</a></li>
   <li><a href=&amp;quot;delete_lst.php&amp;quot;>Usuń  wiadomość</a></li>
  </ul>
 </div>
<?php
 }
?>
 <hr />
 <p  style=&amp;quot;text-align:center&amp;quot;>
  Nasze przedsiębiorstwo &amp;copy;  2009
 </p>
 </div>
```

```
</body>

</html>
```

Skrypt main.php, przedstawiony na listingu 13.3, inicjalizuje sesję oraz sprawdza poprawność danych logowania przekazanych ze strony logowania (index.html). Jeśli dane logowania są niepoprawne, zostajemy przekierowani na stronę logowania. Zauważ, że hasło podane w formularzu logowania, oprócz standardowej filtracji, kodowane jest za pomocą funkcji md5(), a potem dopiero używane w zapytaniu. W bazie danych (dokładniej: w tabeli uzytkownicy) może istnieć tylko jeden użytkownik o danej nazwie i haśle, stąd jeśli zapytanie zwróci jeden wiersz, możemy być pewni, że mamy do czynienia z użytkownikiem uprawnionym do ingerencji w naszą bazę.

Po zalogowaniu użytkownik widzi trzy opcje (łącza) pozwalające na przejście do dodawania danych, edycji i usuwania.

Zacznijmy od dodawania. Listing 13.4 zawiera kod skryptu add.php realizującego formularz dodawania nowej wiadomości.

Listing 13.4. Dodawanie wiadomości — formularz (add.php)

```
<?php

  session_start();

?>

<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.01//EN&quot;

   &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>

<html lang=&quot;pl&quot;>

<head>

  <meta http-equiv=&quot;Content-Type&quot; content=&quot;text/html; charset=utf-8&quot;>

  <title>System zarządzania aktualnościami</title>

</head>

<body>

  <h2>System zarządzania aktualnościami</h2>

  <div>

  <hr />

<?php

  if (!isset($_SESSION['valid_user'])) {

    session_destroy();

    header(&quot;Location: index.html&quot;);
```

```
 }
 else {
?>
 <p>
  Użytkownik: <b><?=  $_SESSION['user_name']; ?></b> [<a
href=&quot;main.php&quot;>Powrót</a>] [<a  href=&quot;logout.php&quot;>Wyloguj</a>]
 </p>
 <hr />
 <h3>Dodawanie wiadomości</h3>
 <form action=&quot;addtodb.php&quot;  method=&quot;POST&quot;>
  <div>
  Tytuł:<br />
  <input name=&quot;tytul&quot;  /><br />
  Treść:<br />
  <textarea cols=&quot;50&quot;  rows=&quot;10&quot; name=&quot;tresc&quot;></textarea>
  <br />
  <input type=&quot;submit&quot;  value=&quot;Dodaj&quot; name=&quot;add&quot; />
  </div>
 </form>
<?php
 }
?>
 <hr />
 <p  style=&quot;text-align:center&quot;>
  Nasze przedsiębiorstwo &copy;  2009
 </p>
 </div>
</body>
```

```
</html>
```

Każdy ze skryptów prezentowanych w tym punkcie sprawdza, czy ustawiona została zmienna sesji o nazwie valid_user — istnienie tej zmiennej oznacza, że zalogowany jest uprawniony użytkownik. Po wprowadzeniu danych do formularza użytkownik zostaje przekierowany do strony addtodb.php (listing 13.5), której kod wykonuje operację dodawania danych do bazy.

Listing 13.5. Dodawanie wiadomości — operacje na bazie danych (addtodb.php)

```
<?php

  session_start();

?>

<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.01//EN&quot;

   &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>

<html lang=&quot;pl&quot;>

<head>

  <meta http-equiv=&quot;Content-Type&quot; content=&quot;text/html; charset=utf-8&quot;>

  <title>System zarządzania aktualnościami</title>

</head>

<body>

  <h2>System zarządzania aktualnościami</h2>

  <div>

  <hr />

<?php

  if (isset($_SESSION['valid_user'])) {

    if (isset($_POST['add'])) {

      $data = date('Y-m-d');

      $tytul = addslashes(strip_tags(trim($_POST['tytul'])));

      $tresc = addslashes(strip_tags(trim($_POST['tresc'])));

      $sql = &quot;INSERT INTO aktualnosci VALUES (nextval('aktualnosci_sq'::regclass), '$tytul', '$tresc',
'$data')&quot;;

      $conn = pg_connect(&quot;dbname=SerwisWWW user=postgres password=test&quot;);

      pg_query($conn, $sql);
```

```
      pg_close($conn);

      print(&quot;Wiadomość została  dodana do bazy.<br />&quot;);

      print(&quot;<a href=&quot;main.php&quot;>Powrót  do strony głównej</a>&quot;);

    } else {

      print(&quot;Nie tędy droga!<br />&quot;);

      print(&quot;<a href=&quot;main.php&quot;>Powrót  do strony głównej</a>&quot;);

    }

  } else {

    session_destroy();

    header(&quot;Location: index.html&quot;);

  }

?>

  </div>

</body>

</html>
```

Dane przed wpisaniem do tabeli są poddawane filtracji w celu uniknięcia iniekcji SQL (listing 13.5). Jeśli użytkownik zostanie skierowany na stronę addtodb.php z innej strony niż ta właściwa, zawierająca formularz dodawania danych, zostanie przekierowany do strony głównej systemu.

Teraz przyjrzyjmy się skryptom edycji. Na początek skrypt, do którego użytkownik zostanie przekierowany ze strony głównej systemu. Skrypt ten, umieszczony w pliku edit_lst.php, przedstawiony został na listingu 13.6.

Listing 13.6. Edycja wiadomości — lista wiadomości z bazy danych (edit_lst.php)

```
<?php

  session_start();

?>

<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD  HTML 4.01//EN&quot;

  &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>

<html lang=&quot;pl&quot;>

<head>

  <meta  http-equiv=&quot;Content-Type&quot; content=&quot;text/html;  charset=utf-8&quot;>
```

```
  <title>System zarządzania  aktualnościami</title>
</head>
<body>
  <h2>System zarządzania aktualnościami</h2>
  <div>
  <hr />
<?php
  if (!isset($_SESSION['valid_user'])) {
      session_destroy();
      header(&quot;Location:  index.html&quot;);
  }
  else {
?>
  <p>
    Użytkownik: <b><?=  $_SESSION['user_name']; ?></b> [
    <a  href=&quot;main.php&quot;>Powrót</a>] [<a  href=&quot;logout.php&quot;>Wyloguj</a>]
  </p>
  <hr />
  <h3>Edycja wiadomości</h3>
  <p>Wybierz wiadomość do edycji:</p>
  <ul>
<?php
  $conn = pg_connect(&quot;dbname=SerwisWWW  user=postgres password=test&quot;);
  $sql = &quot;SELECT * FROM aktualnosci  ORDER BY aktualnosci_data DESC&quot;;
  $res = pg_query($conn, $sql);
  while ($wiersz = pg_fetch_assoc($res)) {
    $id = $wiersz['aktualnosci_id'];
    $tytul = stripslashes($wiersz['aktualnosci_tytul']);
```

```
    $tresc = stripslashes($wiersz['aktualnosci_tresc']);

    $data = $wiersz['aktualnosci_data'];

    $element = &quot;<li><b>$tytul</b><br  />$tresc<br /><i>$data</i><br /><a
href=&quot;edit.php?id=$id&quot;>Edytuj</a></li>&quot;;

    print($element);

  }

  pg_close($conn);

?>

  </ul>

<?php

  }

?>

  <hr />

  <p  style=&quot;text-align:center&quot;>

    Nasze przedsiębiorstwo &copy;  2009

  </p>

  </div>

</body>

</html>
```

Skrypt edit_lst.php (listing 13.6) łączy się z bazą danych i wypisuje wszystkie wiadomości w kolejności od najnowszej do najstarszej. Pod każdą wiadomością umieszczany jest link do strony generującej formularz edycji. W adresie każdego z tych łączy przekazywana jest, jako parametr, wartość pola klucza głównego tabeli aktualnosci. Wartość ta pozwoli jednoznacznie zidentyfikować przez stronę formularza edycji wybraną wiadomość. Po wybraniu jednego z łączy użytkownik skierowany zostaje do strony formularza edycji wybranej wiadomości (edit.php — listing 13.7).

Listing 13.7. Edycja wiadomości — formularz (edit.php)

```
<?php

  session_start();

?>

<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD  HTML 4.01//EN&quot;

  &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>
```

```
<html lang=&quot;pl&quot;>

<head>

  <meta  http-equiv=&quot;Content-Type&quot; content=&quot;text/html;  charset=utf-8&quot;>

  <title>System zarządzania  aktualnościami</title>

</head>

<body>

  <h2>System zarządzania aktualnościami</h2>

  <div>

  <hr />

<?php

  if (!isset($_SESSION['valid_user'])) {

      session_destroy();

      header(&quot;Location:  index.html&quot;);

  }

  else if (!isset($_GET['id'])) {

    header(&quot;Location: main.php&quot;);

  } else {

    $id = addslashes($_GET['id']);

    $conn = pg_connect(&quot;dbname=SerwisWWW  user=postgres password=test&quot;);

    $sql = &quot;SELECT aktualnosci_tytul,  aktualnosci_tresc FROM aktualnosci WHERE
aktualnosci_id=$id&quot;;

    $res = pg_query($conn, $sql);

    $wiersz = pg_fetch_assoc($res);

    pg_close($conn);

?>

  <p>

    Użytkownik: <b><?= $_SESSION['user_name'];  ?></b> [<a
href=&quot;edit_lst.php&quot;>Powrót</a>] [<a  href=&quot;logout.php&quot;>Wyloguj</a>]

  </p>
```

```
  <hr />

  <h3>Edycja wiadomości</h3>

  <form action=&quot;edittodb.php&quot;  method=&quot;POST&quot;>

    <div>

    Tytuł:<br />

    <input name=&quot;tytul&quot;  value=&quot;<?= stripslashes($wiersz['aktualnosci_tytul']); ?>&quot;  /><br />

    Treść:<br />

    <textarea cols=&quot;50&quot;  rows=&quot;10&quot; name=&quot;tresc&quot;><?=
stripslashes($wiersz['aktualnosci_tresc']); ?></textarea>

    <br />

    <input type=&quot;hidden&quot;  name=&quot;id&quot; value=&quot;<?= $id; ?>&quot; />

    <input type=&quot;submit&quot;  value=&quot;Aktualizuj&quot; name=&quot;actual&quot; />

    </div>

  </form>

<?php

  }

?>

  <hr />

  <p  style=&quot;text-align:center&quot;>

    Nasze przedsiębiorstwo &copy;  2009

  </p>

  </div>

</body>

</html>
```

Edycja danych (listing 13.7) przypomina trochę dodawanie wiadomości, tyle że w przypadku edycji najpierw pobierana jest wiadomość do edycji. Data wiadomości nie podlega edycji. Po dokonaniu zmian w treści bądź tytule wiadomości zostajemy przekierowani do skryptu edittodb.php (listing 13.8), który wykonuje operację aktualizacji wiadomości.

Listing 13.8. Edycja wiadomości — operacje na bazie danych (edittodb.php)

```
<?php
```

```
  session_start();

?>

<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD  HTML 4.01//EN&quot;

   &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>

<html lang=&quot;pl&quot;>

<head>

  <meta  http-equiv=&quot;Content-Type&quot; content=&quot;text/html;  charset=utf-8&quot;>

  <title>System zarządzania  aktualnościami</title>

</head>

<body>

  <h2>System zarządzania aktualnościami</h2>

  <div>

  <hr />

<?php

  if (isset($_SESSION['valid_user'])) {

    if (isset($_POST['actual'])) {

      $id = $_POST['id'];

      $tytul =  addslashes(strip_tags(trim($_POST['tytul'])));

      $tresc =  addslashes(strip_tags(trim($_POST['tresc'])));

      $sql = &quot;UPDATE aktualnosci SET  aktualnosci_tytul='$tytul', aktualnosci_tresc='$tresc' WHERE
aktualnosci_id=$id&quot;;

      $conn =  pg_connect(&quot;dbname=SerwisWWW user=postgres password=test&quot;);

      pg_query($conn, $sql);

      pg_close($conn);

      print(&quot;Wiadomość została  zaktualizowana.<br />&quot;);

      print(&quot;<a href=&quot;edit_lst.php&quot;>Powrót  do strony edycji wiadomości</a>&quot;);

    } else {

      print(&quot;Nie tędy droga!<br />&quot;);
```

```
      print(&quot;<a href=&quot;main.php&quot;>Powrót  do strony głównej</a>&quot;);

    }

  } else {

    session_destroy();

    header(&quot;Location: index.html&quot;);

  }

?>

  <hr />

  <p  style=&quot;text-align:center&quot;>

    Nasze przedsiębiorstwo &copy;  2009

  </p>

  </div>

</body>

</html>
```

Skrypt z listingu 13.8 wykonuje aktualizację wiadomości na podstawie danych z formularza edycji. Działanie skryptu jest podobne jak w przypadku addtodb.php (listing 13.5).

Pozostaje jeszcze część systemu odpowiedzialna za usuwanie wiadomości. Listing 13.9 przedstawia skrypt delete_lst.php wyświetlający listę wiadomości, pod którymi znajdują się łącza przekierowujące do skryptu delete.php.

Listing 13.9. Usuwanie wiadomości (delete_lst.php)

```
<?php

  session_start();

?>

<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD  HTML 4.01//EN&quot;

   &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>

<html lang=&quot;pl&quot;>

<head>

  <meta  http-equiv=&quot;Content-Type&quot; content=&quot;text/html; charset=utf-8&quot;>

  <title>System zarządzania  aktualnościami</title>
```

```
</head>

<body>

  <h2>System zarządzania aktualnościami</h2>

  <div>

  <hr />

<?php

  if (!isset($_SESSION['valid_user'])) {

    session_destroy();

    header(&quot;Location: index.html&quot;);

  }

  else {

?>

  <p>

    Użytkownik: <b><?=  $_SESSION['user_name']; ?></b> [

    <a  href=&quot;main.php&quot;>Powrót</a>] [<a  href=&quot;logout.php&quot;>Wyloguj</a>]

  </p>

  <hr />

  <h3>Usuwanie wiadomości</h3>

  <p>Wybierz wiadomość do usunięcia:</p>

  <ul>

<?php

  $conn = pg_connect(&quot;dbname=SerwisWWW  user=postgres password=test&quot;);

  $sql = &quot;SELECT * FROM aktualnosci  ORDER BY aktualnosci_data DESC&quot;;

  $res = pg_query($conn, $sql);

  while ($wiersz = pg_fetch_assoc($res)) {

    $id = $wiersz['aktualnosci_id'];

    $tytul = stripslashes($wiersz['aktualnosci_tytul']);

    $tresc = stripslashes($wiersz['aktualnosci_tresc']);
```

```
    $data = $wiersz['aktualnosci_data'];

    $element =  &quot;<li><b>$tytul</b><br />$tresc<br  /><i>$data</i><br /><a
href=&quot;delete.php?id=$id&quot;>Usuń</a></li>&quot;;

    print($element);

  }

  pg_close($conn);

?>

  </ul>

<?php

  }

?>

  <hr />

  <p  style=&quot;text-align:center&quot;>

    Nasze przedsiębiorstwo &copy;  2009

  </p>

  </div>

</body>

</html>
```

Tak jak w przypadku skryptu edit_lst.php łącze dodawane do każdej wiadomości, a przekierowujące do skryptu usuwania wiadomości z bazy, zaopatrzone jest w parametr pozwalający na jednoznaczną identyfikację wiadomości do usunięcia. Listing 13.10 przedstawia skrypt wykonujący usuwanie danych z bazy.

Listing 13.10. Usuwanie danych — operacje na bazie danych (delete.php)

```
<?php

  session_start();

?>

<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD  HTML 4.01//EN&quot;

   &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>

<html lang=&quot;pl&quot;>

<head>
```

```
  <meta  http-equiv=&quot;Content-Type&quot; content=&quot;text/html;  charset=utf-8&quot;>
  <title>System zarządzania  aktualnościami</title>
</head>
<body>
  <h2>System zarządzania aktualnościami</h2>
  <div>
  <hr />
<?php
  if (isset($_SESSION['valid_user'])) {
    if (isset($_GET['id'])) {
      $id = $_GET['id'];
      $sql = &quot;DELETE FROM aktualnosci  WHERE aktualnosci_id=$id&quot;;
      $conn =  pg_connect(&quot;dbname=SerwisWWW user=postgres password=test&quot;);
      pg_query($conn, $sql);
      pg_close($conn);
      print(&quot;Wiadomość została  usunięta.<br />&quot;);
      print(&quot;<a  href=&quot;delete_lst.php&quot;>Powrót do strony usuwania  wiadomości</a>&quot;);
    } else {
      print(&quot;Nie tędy droga!<br />&quot;);
      print(&quot;<a href=&quot;main.php&quot;>Powrót  do strony głównej</a>&quot;);
    }
  } else {
    session_destroy();
    header(&quot;Location: index.html&quot;);
  }
?>
  <hr />
  <p style=&quot;text-align:center&quot;>
```

```
    Nasze przedsiębiorstwo &copy;  2009
  </p>
  </div>
</body>
</html>
```

Na koniec pozostaje tylko skrypt wylogowujący użytkownika — skrypt ten został przedstawiony na listingu 13.11.

Listing 13.11. Wylogowanie użytkownika (logout.php)

```
<?php
  session_start();
?>
<!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD  HTML 4.01//EN&quot;
   &quot;http://www.w3.org/TR/html4/strict.dtd&quot;>
<html lang=&quot;pl&quot;>
<head>
  <meta  http-equiv=&quot;Content-Type&quot; content=&quot;text/html;  charset=utf-8&quot;>
  <title>System zarządzania  aktualnościami</title>
</head>
<body>
  <h2>System zarządzania aktualnościami</h2>
  <div>
  <hr />
<?php
  if (isset($_SESSION['valid_user'])) {
    unset($_SESSION['valid_user']);
    unset($_SESSION['user_name']);
    session_destroy();
    print(&quot;Nastąpiło poprawne wylogowanie.<br  />&quot;);
```

```
    print(&quot;<a href=&quot;index.html&quot;>Powrót  do strony logowania</a>&quot;);
  } else {
    session_destroy();
    header(&quot;Location: index.html&quot;);
  }
?>
  <hr />
  <p  style=&quot;text-align:center&quot;>
    Nasze przedsiębiorstwo &copy;  2009
  </p>
  </div>
</body>
</html>
```

Skrypt ten usuwa dane sesji, dzięki czemu nie można już wykonywać żadnych operacji na wiadomościach bez wcześniejszego zalogowania się.

Mamy już wszystkie skrypty, sprawdźmy więc, jak działa nasza aplikacja.

Sprawdzenie działania systemu

Najpierw „wejdziemy” na stronę logowania (idex.html), gdzie podamy nazwę użytkownika i hasło (rysunek 13.1).

Rysunek 13.1. Strona logowania

Po zalogowaniu się zostaniemy skierowani na stronę główną systemu generowaną przez skrypt main.php (rysunek 13.2).

Rysunek 13.2. Strona główna systemu zarządzania wiadomościami

Sprawdzimy teraz po kolei każdą z opcji — na początek dodawanie wiadomości (rysunek 13.3). Musimy pamiętać, że data zostanie wygenerowana przez skrypt na podstawie aktualnej daty systemowej serwera.

Rysunek 13.3. Dodawanie wiadomości

Po kliknięciu przycisku Dodaj wiadomość zostanie dopisana do bazy danych, po czym zostaniemy przekierowani do strony głównej systemu.

Następną opcją dostępną na stronie głównej (rysunek 13.2) jest edycja wiadomości. Po kliknięciu tego łącza zostaniemy przeniesieni na stronę, na której wypisane zostały wszystkie wiadomości z bazy (rysunek 13.4).

Rysunek 13.4. Lista wiadomości do edycji

Na liście wiadomości do edycji widzimy również dodaną przed chwilą nową wiadomość. Kliknijmy więc łącze Edytuj poniżej tej wiadomości i przejdźmy do formularza edycji (rysunek 13.5). Wprowadźmy parę zmian w naszej wiadomości i kliknijmy przycisk Aktualizuj.

Rysunek 13.5. Edycja wiadomości

Po kliknięciu przycisku Aktualizuj zostaniemy przekierowani do skryptu realizującego aktualizację, a stamtąd na stronę z listą wiadomości do edycji. Klikając łącze Powrót, znajdziemy się znów na stronie głównej systemu.

Ostatnia opcja dostępna na stronie głównej to usuwanie wiadomości — wybierzmy więc tę opcję. Zostaniemy przeniesieni na stronę zawierającą listę wiadomości do usunięcia (rysunek 13.6).

Rysunek 13.6. Lista wiadomości możliwych do usunięcia

Klikając łącze Usuń umieszczone pod każdą z wiadomości, możemy usunąć wiadomość — zostaniemy przekierowani do skryptu usuwającego wiadomość z bazy.

Jeśli jednak niczego nie usuniemy, poprawnie się wylogujemy (klikając Wyloguj) i uruchomimy aplikację z rozdziału 10., zobaczymy efekt jak na rysunku 13.7 — na samej górze widoczna będzie nowa wiadomość, którą dodaliśmy i edytowaliśmy z użyciem naszego systemu zarządzania wiadomościami.

Rysunek 13.7. Efekt działania systemu zarządzania wiadomościami

Podsumowanie

I mamy gotową aplikację WWW! Oczywiście rozwiązania w niej zastosowane nie stanowią jedynego możliwego wzorca programowania w PHP — są tylko propozycją. Zadaniem dla ambitnych może być rozbudowa tego serwisu. Należy przede wszystkim dopisać wszystkie elementy pominięte, które utrudniłyby analizę tego przykładu, a które powinny się znaleźć w każdej aplikacji WWW działającej w warunkach produkcyjnych. Chodzi

przede wszystkim o sprawdzanie, czy udało się nawiązać połączenie z systemem baz danych oraz czy udało się poprawnie wykonać zapytanie. Dalej trzeba by zadbać o walidację danych wprowadzanych przez użytkownika i wyeliminować sytuację, w której wprowadza się do bazy danych puste ciągi znaków. Warto również pytać użytkownika o potwierdzenie usuwania danych. Wreszcie można całkowicie przebudować ten serwis np. zgodnie z paradygmatem obiektowym, z uwzględnieniem podziału na moduły, z zastosowaniem szablonów, z dodaniem możliwości tworzenia i usuwania kont użytkowników, z kategoryzacją użytkowników… Słowem, można wypróbować wszystko to, co wyjaśniono w trakcie tego kursu.