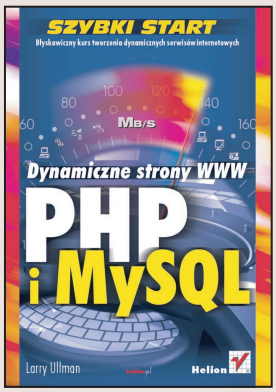
PHP i MySQL. Dynamiczne strony WWW. Szybki start



Autor: Larry Ullman
Tłumaczenie: Michał Dadan (rozdz. 1 – 7),
Piotr Pilch (rozdz. 8 – 13, dod. A – D)
ISBN: 83-7361-157-6
Tytuł oryginału: PHP and MySQL for Dynamic Web Sites VQPG
Format: B5, stron: 580
Przykłady na ftp: 352 kB

Coraz więcej serwisów internetowych składa się nie tylko z atrakcyjnego interfejsu użytkownika, ale także z rozbudowanych aplikacji działających na serwerze. Są one najczęściej oparte o bazy danych, które przechowują dane i zapewniają szybki do nich dostęp. Jeśli chcesz w krótkim czasie nauczyć się tworzyć takie aplikacje, znalazłeś właściwą książkę.

„PHP i MySQL. Dynamiczne strony WWW" nauczy Cię tworzenia dynamicznych serwisów internetowych z wykorzystaniem PHP i MySQL-a. Książka przekaże Ci wiedzę niezbędną dla projektantów rozwiązań internetowych. Wszystkie zagadnienia opisane są „krok po kroku", każdemu z nich towarzyszy też odpowiedni rysunek. Przykłady odzwierciedlają problemy, z którymi projektanci stron internetowych spotykają się na co dzień.

Książka opisuje:

- Podstawy programowania w PHP
- Tworzenie dynamicznych stron internetowych z użyciem PHP
- Zasady projektowanie baz danych
- Język SQL
- Korzystanie z systemu zarządzania bazami danych MySQL
- Łączenie PHP z systemem MySQL
- Użycie sesji
- Zabezpieczanie stron internetowych przed dostępem nieuprawnionych osób
- Przykładowe aplikacje: rejestracja użytkowników i sklep internetowy

Treść książki uzupełniają dodatki opisujące sposób instalacji omawianych w niej narzędzi oraz dodatkowe, przydatne aplikacje.

Jeśli chcesz w szybko nauczyć się programowania dynamicznych serwisów internetowych, ta książka będzie Twoim intensywnym kursem. Już po przeczytaniu kilku rozdziałów będziesz w stanie pisać pierwsze programy w PHP korzystające z bazy MySQL, a po przeczytaniu całej książki poradzisz sobie również z tworzeniem bardziej rozbudowanych aplikacji.

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

Spis treści

Zarządzanie zawartością strony — przykład

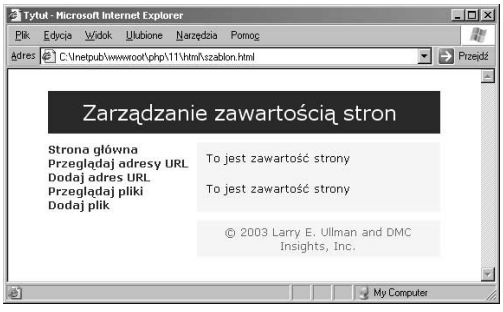
11

Pierwszy przykład aplikacji zamieszczony w niniejszej książce będzie dotyczył zarządzania zawartością strony. Aplikacja będzie zarządzała zarówno adresami URL, jak i plikami, które użytkownicy będą mogli dodawać, wyświetlać i przetwarzać. Co prawda w tym przypadku nie zostanie uwzględniona administracyjna część aplikacji, ale zostaną dołączone informacje pozwalające na jej wykonanie.

Chociaż w tym rozdziale skupimy się na konkretnym przykładzie, to jednak wspomnimy również o kilku nowych funkcjach i metodach. Do funkcji tych należy zaliczyć `mysql_insert_id()` i `list()`. Wspomnimy także, w jaki sposób zrealizować operację umieszczania plików na serwerze.

Tworzenie szablonu

W pierwszym etapie projektowania aplikacji zostanie utworzony systemowy szablon, którego zadaniem będzie wspomaganie procesu przygotowywania strony HTML. Strona w ostatecznej postaci (rysunek 11.1) będzie korzystała z tabel i kilku arkuszy stylów CSS (ang. *Cascading Style Sheets*).



Rysunek 11.1. *Domyślny wygląd strony aplikacji omawianej w tym rozdziale*

Aby utworzyć plik naglowek.html, należy wykonać następujące kroki:

1. W edytorze tekstu utworzyć nową stronę HTML (listing 11.1).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD
➥XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/2000/
         ➥REC-xhtml1-20000126/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>
```

1. Napisać kod arkusza stylów CSS.

```
<style type="text/css" media="screen">
body { background-color: #ffffff; }

.content {
  background-color: #f5f5f5;
  padding-top: 10px; padding-right: 10px;
  ➥padding-bottom: 10px; padding-left: 10px;
  ➥margin-top: 10px; margin-right: 10px;
  ➥margin-bottom: 10px; margin-left: 10px;
}
```

Listing 11.1. *Plik naglowek.html rozpoczyna się od kodu formatującego stronę HTML i zawiera wymagany kod arkusza stylów CSS*

Listing

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/2000/REC-xhtml1-20000126/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="screen">
body { background-color: #ffffff; }
.content {
  background-color: #f5f5f5;
  padding-top: 10px; padding-right: 10px; padding-bottom: 10px; padding-left: 10px;
  margin-top: 10px; margin-right: 10px; margin-bottom: 10px; margin-left: 10px;
}
a.navlink:link { color: #003366; text-decoration: none; }
a.navlink:visited { color: #003366; text-decoration: none; }
a.navlink:hover { color: #cccccc; text-decoration: none; }
td {
  font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 13px;
  vertical-align: top;
}
.title {
  font-size: 24px; font-weight: normal; color: #ffffff;
  margin-top: 5px; margin-bottom: 5px; margin-left: 20px;
  padding-top: 5px; padding-bottom: 5px; padding-left: 20px;
}
</style>
</head>
<body>

<table width="90%" border="0" cellspacing="10" cellpadding="0"
align="center">
<tr>
  <td colspan="2" bgcolor="#003366"><p class="title">Zarządzanie zawartością stron</p></td>
</tr>
<tr>
  <td valign="top" nowrap="nowrap">
<b><a href="index.php" class="navlink">Strona główna</a><br />
<a href="view_urls.php" class="navlink">Przeglądaj adresy URL</a><br />
<a href="add_url.php" class="navlink">Dodaj adres URL</a><br />
<a href="view_files.php" class="navlink">Przeglądaj pliki</a><br />
<a href="add_file.php" class="navlink">Dodaj plik</a></b>
  </td>

  <td valign="top" class="content">
  <!-- Listing 11.1 - naglowek.html -->
```

```
a.navlink:link { color: #003366;
➥text-decoration: none; }
a.navlink:visited { color: #003366;
➥text-decoration: none; }
a.navlink:hover { color: #cccccc;
➥text-decoration: none; }
td {
  font-family: Verdana, Arial, Helvetica,
  ➥sans-serif; font-size: 13px;
  vertical-align: top;
}
.title {
  font-size: 24px; font-weight:
  ➥normal; color: #ffffff;
  margin-top: 5px; margin-bottom: 5px;
  ➥margin-left: 20px;
  padding-top: 5px; padding-bottom: 5px;
  ➥padding-left: 20px;
}
</style>
```

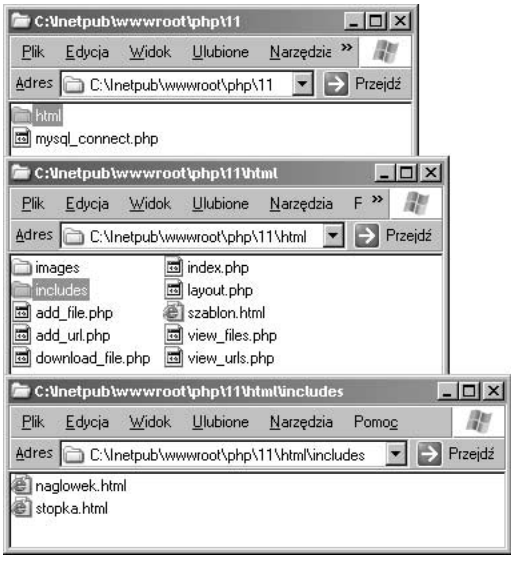
W przypadku omawianej aplikacji zostanie wykorzystany arkusz stylów CSS, który posłuży do sformatowania tekstu. Ze względu na niewielką objętość kodu arkusza stylów CSS, zamiast w oddzielnym pliku zostanie on umieszczony bezpośrednio w sekcji HEAD strony HTML.

3. Zakończyć pisanie kodu nagłówka strony HTML i utworzyć pierwszy wiersz tytułu.

```
</head>
<body>
<table width="90%" border="0"
➥cellspacing="10" cellpadding="0"
➥align="center">
  <tr>
    <td colspan="2" bgcolor="#003366">
    ➥<p class="title">Zarządzanie
    ➥zawartością stron</p></td>
  </tr>
  <tr>
```

Podstawowa strona HTML będzie zawierała tabelę złożoną z trzech wierszy — tytułu, wiersza przechowującego odnośniki i zawartość strony oraz wiersza, w którym będą umieszczone informacje o prawach autorskich (rysunek 11.2).

Rysunek 11.2. *Wygląd strony, na której w celu wyróżnienia wierszy i kolumn tabeli zastosowano obwódki*



Rysunek 11.3. *Struktura aplikacji internetowej, której katalogiem głównym jest katalog html*

4. Utworzyć sekcję powiązaną z odnośnikami i rozpocząć definiowanie komórek przechowujących zawartość strony.

```
<td valign="top" nowrap="nowrap">
<b><a href="index.php" class=
➥"navlink">Strona główna</a><br />
<a href="view_urls.php"
➥class="navlink">Przeglądaj adresy
➥URL</a><br />
<a href="add_url.php" class=
➥"navlink">➥Dodaj adres URL</a><br />
<a href="view_files.php" class=
➥"navlink">Przeglądaj pliki</a><br />
<a href="add_file.php"
➥class="navlink">Dodaj plik</a></b>
</td>
<td valign="top" class="content">
<!-- Listing 11.1 - naglowek.html -->
```

5. Zapisać plik pod nazwą *naglowek.html* i umieścić go na serwerze WWW (w katalogu *includes*).

Na rysunku 11.3 przedstawiono strukturę katalogów zawierających pliki aplikacji.

Wskazówka

- W przykładzie zamieszczonym w następnym przykładzie zostanie zastosowany bardziej złożony arkusz stylów CSS, który będzie zapisany w oddzielnym pliku dołączonym do pliku *naglowek.html*.

Aby utworzyć plik stopka.html, należy wykonać następujące kroki:

1. W edytorze tekstu utworzyć nową stronę HTML (listing 11.2).

```
<!-- Listing 11.2 - stopka.html -->
```

2. Zakończyć kod środkowego wiersza tabeli.

```
    </td>
  </tr>
```

   Cała zawartość strony będzie przechowywana w środkowym wierszu tabeli, którego definicja zaczyna się w pliku *naglowek.html*. Powyższy kod kończy definicję wiersza, po którym zostanie umieszczony trzeci i ostatni wiersz.

3. Zdefiniować ostatni wiersz i zakończyć tworzenie strony HTML.

```
    <tr>
      <td> </td>
      <td align="center">&copy; 2003 Larry E.
      ➥Ullman and DMC Insights, Inc.</td>
    </tr>
  </table>
  </body>
  </html>
```

4. Zapisać plik pod nazwą *stopka.html* i umieścić go na serwerze WWW (w katalogu *includes*).

Listing 11.2. *Plik stopka.html zawiera pozostałe definicje wyglądu strony HTML*

Listing

```
1   <!-- Listing 11.2 - stopka.html -->
2   <!-- Koniec zawartości strony -->
3       </td>
4     </tr>
5
6     <tr>
7       <td> </td>
8       <td align="center">&copy;
        2003 Larry E.Ullman and DMC
        Insights, Inc.</td>
9     </tr>
10
11    </table>
12    </body>
13    </html>
```

Listing 11.3. *Główna strona aplikacji (w razie potrzeby należy umieścić na niej bardziej wartościowe informacje)*

Listing

```
1   <?php # Listing 11.3 - index.php
2   // Główna strona serwera WWW.
3   // Ustawienie tytułu strony
    i dołączenie nagłówka HTML.
4   $page_title = 'Zarządzanie
    zawartością stron';
5   include_once
    ('includes/naglowek.html');
6   ?>
7   <p>Spam spam spam spam spam spam
    spam spam spam spam.</p>
8   <p>Spam spam spam spam spam spam
    spam spam spam spam.</p>
9   <?php // Dołączenie stopki HTML.
10  include_once ('includes/stopka.html');
11  ?>
```

Rysunek 11.4. *Główna strona*

Tworzenie zwykłych stron internetowych

Zanim zajmiemy się najważniejszą częścią aplikacji internetowej (samym mechanizmem zarządzania zawartością stron) konieczne będzie stworzenie jeszcze dwóch stron HTML. Jedna z nich o nazwie *index.php* będzie spełniała rolę głównej strony aplikacji. Z kolei druga o nazwie *mysql_connect.php* jest skryptem, którego zadaniem jest połączenie z serwerem MySQL i wybranie wymaganej bazy danych (w celu uzyskania dodatkowych informacji na jej temat należy zapoznać się z zawartością ramki *Schemat bazy danych* zawartej w następnym podrozdziale).

Aby utworzyć główną stronę, należy wykonać następujące kroki:

1. W edytorze tekstu utworzyć nowy skrypt PHP (listing 11.3).

```
<?php # Listing 11.3 - index.php
```

2. Określić tytuł strony i dołączyć plik nagłówka HTML.

```
$page_title = 'Zarządzanie zawartością
➥stron';
include_once ('includes/naglowek.html');
?>
```

3. Wypełnić stronę tekstem.

```
<p>Spam spam spam spam spam spam spam
➥spam spam spam.</p>
<p>Spam spam spam spam spam spam spam
➥spam spam spam.</p>
```

   Co prawda posłużyłem się tego typu treścią (słowo `spam`), ale oczywiście na stronie głównej można umieścić bardziej wartościowe informacje.

4. Dołączyć stopkę HTML.

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

5. Zapisać plik pod nazwą *index.php*, umieścić go na serwerze WWW i przetestować przy użyciu przeglądarki internetowej (rysunek 11.4).

Aby utworzyć skrypt mysql_connect.php, należy wykonać następujące kroki:

1. W edytorze tekstu utworzyć nowy skrypt PHP (listing 11.4).

   ```
   <?php # Listing 11.4 - mysql_connect.php
   ```

   Skrypt jest taki sam jak jego poprzednie wersje wykorzystane w poprzednich rozdziałach (z wyjątkiem nazwy użytkownika i jego hasła oraz konfiguracji bazy danych). Skrypt może zostać stworzony od podstaw lub poprzez odpowiednią modyfikację poprzedniej wersji.

2. Określić konfigurację bazy danych w postaci stałych.

   ```
   define ('DB_USER', 'username');
   define ('DB_PASSWORD', 'password');
   define ('DB_HOST', 'localhost');
   define ('DB_NAME', 'content');
   ```

   Mając na względzie kwestię zabezpieczeń użytkownik łączący się z serwerem MySQL powinien na potrzeby omawianej aplikacji dysponować możliwością wykonywania tylko takich poleceń jak `INSERT`, `SELECT`, `DELETE` i dodatkowo wyłącznie dla bazy danych *content*.

3. Połączyć się z serwerem MySQL i wybrać bazę danych.

   ```
   $dbc = @mysql_connect (DB_HOST, DB_USER,
   ➥DB_PASSWORD) OR die ('Nie było możliwe
   ➥połączenie z serwerem MySQL: ' .
   ➥mysql_error() );
   mysql_select_db (DB_NAME) OR die
   ➥('Nie było możliwe wybranie bazy danych: '
   ➥. ➥mysql_error() );
   ```

   Aplikacja dysponuje podstawowym mechanizmem obsługi błędów realizowanym przez funkcję `mysql_error()`. Więcej informacji na ten temat zawarto w rozdziale 6.

4. Zakończyć skrypt PHP.

   ```
   ?>
   ```

5. Zapisać plik pod nazwą *mysql_connect.php* i umieścić go na serwerze WWW poza katalogiem głównym (rysunek 11.3).

Listing 11.4. *Skrypt łączy z serwerem MySQL i wybiera bazę danych content*

Listing

```
1   <?php # Listing 11.4 -mysql_connect.php
2   // W pliku zawarto parametry
    wymagane do uzyskania dostępu
    do bazy danych. Plik jest
    używany także przy łączeniu się
    z serwerem MySQL i wybieraniem
    bazy danych.
4   // Ustawienie jako stałych
    parametrów udzielających dostępu
    do bazy danych.
5   define ('DB_USER', 'username');
6   define ('DB_PASSWORD',
    'password');
7   define ('DB_HOST', 'localhost');
8   define ('DB_NAME', 'content');
9   // Nawiązanie połączenia
    z serwerem i wybranie bazy danych.
10  $dbc = @mysql_connect (DB_HOST,
    DB_USER, DB_PASSWORD) OR die
    ('Nie było możliwe połączenie
    z serwerem MySQL: ' . mysql_error() );
11  mysql_select_db (DB_NAME) OR die
    ('Nie było możliwe wybranie bazy
    danych: ' . mysql_error() );
12  ?>
```

Wskazówki

- W następnym rozdziale zostanie omówiony przykład, w którym zostaną wykorzystane bardziej zaawansowane metody obsługi błędów. Poza tym zamieszczono w nim zmodyfikowaną wersję skryptu *mysql_connect.php*.
- W celu uzyskania dodatkowych informacji na temat praw dostępu do serwera MySQL należy zapoznać się z zawartością dodatku A.

Zarządzanie adresami URL

Kolejne dwie strony HTML, które zostaną stworzone, umożliwią użytkownikom dodawanie adresów URL i przeglądanie już dołączonych. Ta część aplikacji bazuje na trzech tabelach bazy danych — `urls`, `url_types` i `url_titles`. Aby uzyskać dodatkowe informacje, należy zapoznać się z zawartością ramki *Schemat bazy danych*.

Dodawanie adresów URL

Skrypt *add_url.php* prawdopodobnie jest najbardziej złożonym wśród wszystkich wchodzących w skład aplikacji. Wynika to z zastosowania znormalizowanej struktury bazy danych. Formularz zawarty w skrypcie pobiera adres URL, jego tytuł (lub nazwę), opis i maksymalnie trzy kategorie. Po wypełnieniu formularza (adres URL, jego tytuł, opis) dane zostaną umieszczone w tabeli `url_titles`. Następnie w celu dodania do tabeli `urls` od jednego do trzech rekordów (w zależności od ilości kategorii wybranych przez użytkownika) zostanie zastosowany klucz główny tabeli `title_id` wraz z wartościami pola `type_id`.

W celu określenia wartości pola `title_id` dla dodanego adresu URL zostanie wykorzystana funkcja `mysql_insert_id()`, która do tej pory nie była omawiana. Każdorazowo, gdy na tabeli zawierającej pole o automatycznie zwiększanej wartości (zazwyczaj jest nim klucz główny) zostanie wykonane zapytanie, wtedy dla pola serwer MySQL użyje następnej wartości. Funkcja `mysql_insert_id()` zwróci tę wartość. W przeciwnym razie nie byłoby możliwe jej uzyskanie.

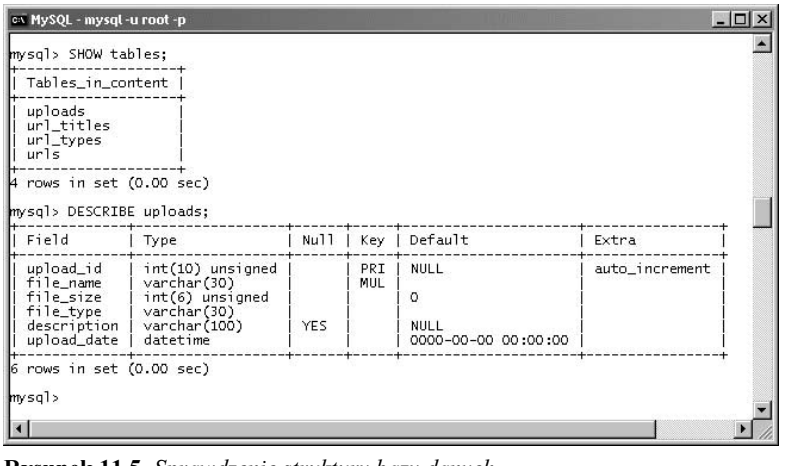
Schemat bazy danych

W niniejszym rozdziale zastosowano bazę danych o nazwie *content*, którą utworzono w rozdziale 5. W tamtym rozdziale zdefiniowano trzy znormalizowane tabele na potrzeby obsługi adresów URL (złożona struktura pozwala na zaklasyfikowanie adresów URL do wielu typów). W tym rozdziale zostanie stworzona tabela `uploads` wykorzystywana przy zarządzaniu plikami. Całkowita struktura bazy danych może zostać ponownie zdefiniowana z wykorzystaniem poniższych poleceń:

```
CREATE TABLE uploads (upload_id int (10) UNSIGNED NOT NULL AUTO_INCREMENT,
➥file_name VARCHAR(30) NOT NULL, file_size INT(6) UNSIGNED NOT NULL, file_type
➥VARCHAR(30) NOT NULL, description VARCHAR(100) DEFAULT NULL, upload_date
➥DATETIME NOT NULL, PRIMARY KEY (upload_id), KEY file_name (file_name));
CREATE TABLE url_titles (title_id SMALLINT(4) UNSIGNED NOT NULL AUTO_INCREMENT,
➥url VARCHAR(60) NOT NULL, title VARCHAR(60) NOT NULL, description TINYTEXT NOT
➥NULL, PRIMARY KEY (title_id), UNIQUE KEY url (url));
CREATE TABLE url_types (type_id TINYINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
➥type VARCHAR(20) NOT NULL, PRIMARY KEY (type_id), UNIQUE KEY type (type));
CREATE TABLE urls (url_id SMALLINT(4) UNSIGNED NOT NULL AUTO_INCREMENT,
➥title_id SMALLINT(4) UNSIGNED NOT NULL, type_id TINYINT(3) UNSIGNED NOT NULL,
➥approved CHAR(1) DEFAULT 'N', date_submitted TIMESTAMP(14) NOT NULL, PRIMARY
➥KEY (url_id), KEY title_id (title_id), KEY type_id (type_id), KEY
➥date_submitted (date_submitted));
```

W celu sprawdzenia struktury bazy danych zawsze można użyć poniższych poleceń języka SQL (rysunek 11.5).

```
SHOW TABLES;
DESCRIBE nazwa_tabeli;
```



Rysunek 11.5. *Sprawdzenie struktury bazy danych*

Aby utworzyć skrypt add_url.php, należy wykonać następujące kroki:

1. W edytorze tekstu utworzyć nowy skrypt PHP (listing 11.5).

```
<?php # Listing 11.5 - add_url.php
$page_title = 'Dodawanie adresów URL';
include ('includes/naglowek.html');
require_once ('../mysql_connect.php');
```

2. Sprawdzić, czy formularz został wypełniony i zastosować funkcję `escape_data()`.

```
if (isset($_POST['submit'])) {
➥// Obsługa formularza.
  function escape_data ($data) {
    global $dbc;
    if (ini_get('magic_quotes_gpc')) {
      $data = stripslashes($data);
    }
    return mysql_real_escape_string
    ➥(trim ($data), $dbc);
  }
```

Podobnie jak w poprzednich rozdziałach, funkcja `escape_data()` gwarantuje poprawność przesyłanych danych niezależnie od tego, czy uaktywniono funkcję *Magic Quotes*, czy nie.

3. Przeprowadzić walidację wprowadzonego adresu URL i jego tytułu.

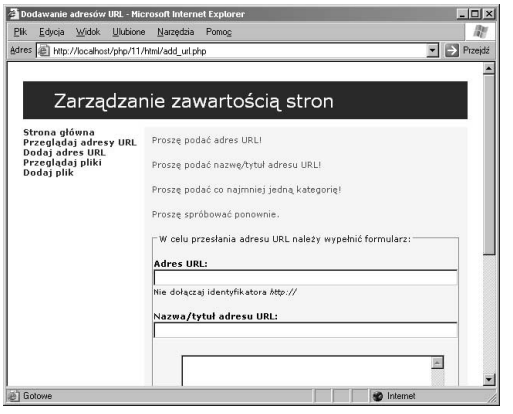
```
if (!empty($_POST['url'])) {
  $u = escape_data($_POST['url']);
} else {
$u = FALSE;
echo '<p><font color="red">Proszę podać
➥adres URL!</font></p>';  }
if (!empty($_POST['title'])) {
  $t = escape_data($_POST['title']);
} else {
  $t = FALSE;
  echo '<p><font color="red">
  ➥Proszę podać nazwę adresu
  ➥URL/title!</font></p>'; }
```

Listing 11.5. *Skrypt pozwala użytkownikom przesyłać adresy URL do bazy danych*

Listing

```
1   <?php # Listing 11.5 - add_url.php
2   // Strona pozwala użytkownikom dodać adresy URL do bazy danych.
3   // Ustawienie tytułu strony i dołączenie nagłówka HTML.
4   $page_title = 'Dodawanie adresów URL';
5   include ('includes/naglowek.html');
6   require_once ('../mysql_connect.php'); // Połączenie z bazą danych.
7   if (isset($_POST['submit'])) { // Obsługa formularza.
8     // Funkcja usuwająca znak '/' i obcinająca dane wprowadzone w formularzu.
9     function escape_data ($data) {
10      global $dbc;
11      if (ini_get('magic_quotes_gpc')) {
12        $data = stripslashes($data);
13      }
14      return mysql_real_escape_string (trim ($data), $dbc);
15    } // Koniec sekcji funkcji escape_data().
```

W tym przykładzie zostanie dokonane sprawdzenie, czy wprowadzono wartości. W celu zwiększenia dokładności można oczywiście zastosować wyrażenia proste (tak naprawdę w rozdziale 8. zawarto skrypt przeprowadzający walidację adresów URL). Jeśli jakieś pole nie zostanie wypełnione, pojawi się komunikat błędu widoczny na rysunku 11.6.



Rysunek 11.6. *Jeśli użytkownik nie poda adresu URL lub jego tytułu, zostanie wyświetlony komunikat błędu*

4. Sprawdzić, czy w polu `description` podano wartość.

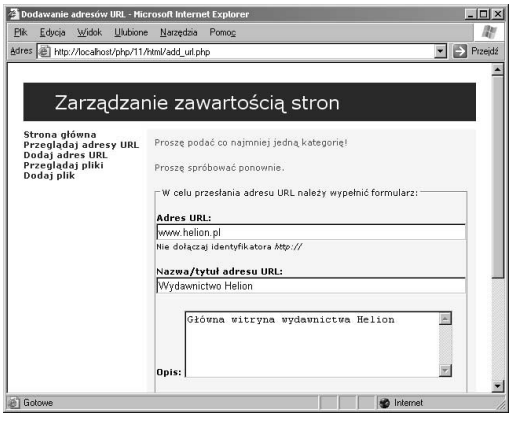
```
if (!empty($_POST['description'])) {
  $d = escape_data
  ➥($_POST['description']);
} else {
  $d = ''; }
```

Ze względu na to, że w polu `description` tabeli `url_titles` nie trzeba wstawiać wartości, będzie ona przetwarzana, ale tylko wtedy, gdy zostanie wprowadzona w formularzu. Jeśli w polu `description` nie zostanie wstawiona żadna wartość, zmiennej `$d` stosowanej w zapytaniu będzie przypisywany pusty łańcuch.

Listing 11.5. *Skrypt pozwala użytkownikom przesyłać adresy URL do bazy danych — ciąg dalszy*

Listing

```
16    // Sprawdzenie adresu URL.
17    if (!empty($_POST['url'])) {
18    $u = escape_data($_POST['url']);
19    } else {
20      $u = FALSE;
21      echo '<p><font color="red">Proszę podać adres URL!</font></p>'; }
22    // Sprawdzenie nazwy adresu URL.
23    if (!empty($_POST['title'])) {
24      $t = escape_data($_POST['title']);
25    } else {
26      $t = FALSE;
27      echo '<p><font color="red">Proszę podać nazwę adresu URL/title!</font></p>';  }
28    // Sprawdzenie wartości pola description (nie jest wymagane).
29    if (!empty($_POST['description'])) {
30      $d = escape_data($_POST['description']);
31    } else {
      $d = ''; }
```



Rysunek 11.7. *Użytkownik musi wybrać co najmniej jedną kategorię dla wprowadzonego adresu URL*

5. Sprawdzić wybraną kategorię.

```
if (($_POST['type1'] > 0) OR
➥($_POST['type2'] > 0) OR
➥($_POST['type3']) > 0) {
  $type = TRUE;
} else {
  $type = FALSE;
  echo '<p><font color="red">
  ➥Proszę podać co najmniej jedną
  ➥kategorię!</font></p>';  }
```

Użytkownik, mając do dyspozycji zbiór typów zawartych w tabeli `url_types`, może przypisać wprowadzanemu adresowi URL maksymalnie trzy kategorie. Zakłada się, że użytkownik wybierze kategorię przy użyciu pierwszego menu rozwijanego (`type1`). Niezależnie od tego instrukcja warunkowa sprawdzi, czy skorzystano z przynajmniej jednego menu rozwijanego. Jeśli tak nie będzie, zostanie wyświetlony komunikat błędu (rysunek 11.7).

Listing 11.5. *Skrypt pozwala użytkownikom przesyłać adresy URL do bazy danych — ciąg dalszy*

Listing

```
32    // Sprawdzenie kategorii.
33    if (($_POST['type1'] > 0) OR ($_POST['type2'] > 0) OR ($_POST['type3']) >0) {
34      $type = TRUE;
35    } else {
36      $type = FALSE;
37      echo '<p><font color="red">Proszę podać co najmniej jedną kategorię!</font></p>'; }
38    if ($u && $t && $type) { // Jeśli wszystko jest w porządku.
39      // Dodawanie adresu URL do tabeli url_titles.
40      $query = "INSERT INTO url_titles (url, title, description) VALUES
        ('$u','$t', '$d')";
41      $result = @mysql_query ($query); // Wykonanie zapytania.
42      $tid = @mysql_insert_id(); // Pobranie identyfikatora tytułu.
43
44      if ($tid > 0) { // Jeśli zostało wykonane prawidłowo.
45        // Tworzenie zapytania.
46        $query = 'INSERT INTO urls (title_id, type_id, approved, date_submitted) VALUES ';
47        if ($_POST['type1'] > 0) {
48          $query .= "($tid, {$_POST['type1']}, 'Y', NOW()), "; }
```

6. Dodać adres URL do tabeli `url_titles`.

```
if ($u && $t && $type) {
  // Dodawanie adresu URL do tabeli
  ➥url_titles.
  $query = "INSERT INTO url_titles (url,
  ➥title, description) VALUES ('$u',
  ➥'$t', '$d')";
  $result = @mysql_query ($query);
  $tid = @mysql_insert_id();
```

Ze względu na strukturę bazy danych, adres URL musi zostać dodany do tabeli `url_titles` zanim będzie możliwa modyfikacja tabeli `urls`. Powyższe zapytanie doda adres URL, a następnie przy użyciu funkcji `mysql_insert_id()` zwróci wartość pola `title_id` (klucz główny tabeli o automatycznie zwiększanej wartości). Uzyskana wartość może następnie zostać wykorzystana w drugim zapytaniu (krok 7.).

7. Utworzyć główne zapytanie.

```
if ($tid > 0) {

  $query = 'INSERT INTO urls
  ➥(title_id, type_id, approved,
  ➥date_submitted) VALUES ';
  if ($_POST['type1'] > 0) {
    $query .= "($tid,
    ➥{$_POST['type1']}, 'Y',
    ➥NOW()), ";
  }
  if ($_POST['type2'] > 0) {
    $query .= "($tid, {$_POST
    ➥['type2']}, 'Y', NOW()), ";
  }
  if ($_POST['type3'] > 0) {
    $query .= "($tid,
    ➥{$_POST['type3']}, 'Y',
    ➥NOW()), ";
  }
  $query = substr ($query, 0, -2);
```

Listing 11.5. *Skrypt pozwala użytkownikom przesyłać adresy URL do bazy danych — ciąg dalszy*

Listing

```
49          if ($_POST['type2'] > 0) {
50            $query .= "($tid, {$_POST['type2']}, 'Y', NOW()), ";          }
51          if ($_POST['type3'] > 0) {
52            $query .= "($tid, {$_POST['type3']}, 'Y', NOW()), ";
53          }
54          $query = substr ($query, 0, -2); // Usunięcie ostatniego przecinka i spacji.
55          // Wykonanie zapytania.
56          $result = @mysql_query ($query);
57          if ($result) {
58            echo '<p><b>Dziękujemy za przesłanie danych!</b></p>';
59            $_POST = array();
60          } else { // Jeśli wystąpią problemy.
61            echo '<p><font color="red">Na skutek błędu systemowego wysłane zapytanie nie mogło
              zostać przetworzone. Przepraszamy za zaistniałą niedogodność.</font></p>';
62          }
63        } else { // Jeśli wystąpią problemy.
64      echo '<p><font color="red">Na skutek błędu systemowego wysłane zapytanie
        nie mogło zostać przetworzone. Przepraszamy za zaistniałą niedogodność.</font></p>';
65      }
```

Jeśli zostanie pobrana wartość zmiennej `$tid`, bez obaw można przejść do wykonywania głównego zapytania (dodanie rekordu do tabeli `urls`). Aby tak postąpić, na początku należy zdefiniować zapytanie i przypisać jego początkową część zmiennej `$query`. Po sprawdzeniu, czy wybrano kategorie (przy użyciu menu rozwijanych), dla każdej z nich dodano rekord przy użyciu zapytania. Na końcu w celu usunięcia z zapytania ostatnich dwóch znaków (przecinek i znak spacji) zostanie użyta funkcja `substr()`. Jeśli wystąpią problemy ze składnią zapytania należy nakazać interpreterowi języka PHP wyświetlenie wartości zmiennej `$query`, dzięki czemu można stwierdzić, co będzie przetwarzane przez serwer MySQL.

8. Wykonać zapytanie, wprowadzić komunikaty informujące o jego wyniku i uzupełnić instrukcje warunkowe.

```
$result = @mysql_query ($query);
if ($result) {
  echo '<p><b>Dziękujemy za
  ➥przesłanie danych!</b></p>';
} else { // Jeśli wystąpią problemy.
  echo '<p><font color="red">
  ➥Na skutek błędu systemowego
  ➥wysłane zapytanie nie mogło
  ➥zostać przetworzone.
  ➥Przepraszamy za zaistniałą
  ➥niedogodność.</font></p>';
}
```

Powyższe instrukcje warunkowe wyświetlą odpowiednie komunikaty związane z przesyłanymi danymi.

Listing 11.5. *Skrypt pozwala użytkownikom przesyłać adresy URL do bazy danych — ciąg dalszy*

Listing

```
66     } else { // Jeśli jedna z operacji sprawdzenia danych się nie powiedzie.
67       echo '<p><font color="red">Proszę spróbować ponownie.</font></p>';
68     }
69   } // Koniec instrukcji warunkowej obsługującej dane wprowadzane w formularzu.
70   // --------- Formularz wyświetlający dane---------
71   // Tworzenie menu rozwijanego.
72   $query = "SELECT * FROM url_types ORDER BY type ASC";
73   $result = @mysql_query ($query);
74   $pulldown = '<option>Wybierz jedną pozycję</option>';
75   while ($row = mysql_fetch_array ($result, MYSQL_ASSOC)) {
76     $pulldown .= "<option value=\"{$row['type_id']}\">{$row['type']}</option>\n";
77   }
78   ?>
79   <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
80   <fieldset><legend>W celu przesłania adresu URL należy wypełnić formularz
     (można wybrać maksymalnie 3 kategorie):</legend>
81   <p><b>Adres URL:</b> <input type="text" name="url" size="60" maxlength="60"
     value="<?php if (isset($_POST['url'])) echo $_POST['url']; ?>" /><br
82     /><small>Nie dołączaj identyfikatora <i>http://</i></small></p>
83    } else {
84    echo '<p><font color="red">Na skutek błędu systemowego wysłane zapytanie
      nie mogło zostać przetworzone. Przepraszamy za zaistniałą niedogodność.</font></p>';
85    }

86    } else {
87    echo '<p><font color="red">Proszę spróbować ponownie.</font></p>';
88    }
89    }
```

9. Utworzyć stronę HTML, na której zostaną umieszczone menu rozwijane umożliwiające wybranie typu adresów URL.

```
$query = "SELECT * FROM url_types ORDER BY
➥type ASC";
$result = @mysql_query ($query);
$pulldown = '<option>Wybierz jedną
➥pozycję</option>
';
while ($row = mysql_fetch_array ($result,
➥MYSQL_ASSOC)) {
  $pulldown .= "<option
  ➥value=\"{$row['type_id']}\">{$row
  ➥['type']}</option>\n";
}
```

Listing 11.5. *Skrypt pozwala użytkownikom przesyłać adresy URL do bazy danych — ciąg dalszy*

Listing

```
90   <p><b>Nazwa/tytuł adresu URL:</b> <input type="text" name="title" size="60"
     maxlength="60" value="<?php if (isset($_POST['title'])) echo $_POST['title']; ?>" /></p>
91
92   <p><b>Opis:</b> <textarea name="description" cols="40" rows="5"><?php if
     (isset($_POST['description'])) echo $_POST['description']; ?></textarea></p>
93
94   <p><b>Kategoria 1:</b> <select name="type1">
95   <?php echo $pulldown; ?>
96   </select></p>
97
98   <p><b>Kategoria 2:</b> <select name="type2">
99   <?php echo $pulldown; ?>
100  </select></p>
101
102  <p><b>Kategoria 3:</b> <select name="type3">
103  <?php echo $pulldown; ?>
104  </select></p>
105
106  </fieldset>
107
108  <div align="center"><input type="submit" name="submit" value="Wyślij" /></div>
109
110  </form><!-- Koniec formularza-->
111  <?php
112  mysql_close(); // Zamknięcie połączenia z bazą danych.
113  include ('includes/stopka.html'); // Dołączenie stopki HTML.
114  ?>
```

Rysunek 11.8. *Kod źródłowy strony HTML dynamicznie generujący menu rozwijane*

Formularz będzie zawierał trzy menu rozwijane, z których każde będzie dysponowało listą dostępnych typów adresów URL pobranych z bazy danych. Ze względu na to, że zależy mi na wyświetleniu trzech kopii tego typu informacji, bardziej efektywnym będzie przypisanie wyników zmiennej niż wielokrotne wysyłanie zapytania do bazy danych.
Na rysunku 11.8 pokazano kod źródłowy strony zawierającej menu rozwijane.

10. Utworzyć formularz na stronie HTML.

```
<form action="<?php echo
$_SERVER['PHP_SELF']; ?>" method="post">
<fieldset><legend>W celu przesłania adresu
➥URL należy wypełnić formularz (można
➥wybrać maksymalnie 3 kategorie):</legend>

<p><b>Adres URL:</b> <input type="text"
➥name="url" size="60" maxlength="60"
➥value="<?php if (isset($_POST['url']))
➥echo $_POST['url']; ?>" /><br
➥/><small>➥Nie dołączaj identyfikatora
➥<i>http://</i></small></p>

<p><b>Nazwa/tytuł adresu URL:</b> <input
➥type="text" name="title" size="60"
➥maxlength="60" value="<?php if
➥(isset($_POST['title'])) echo
➥$_POST['title']; ?>" /></p>

<p><b>Opis:</b> <textarea name="description"
➥cols="40" rows="5"><?php if
➥(isset($_POST['description'])) echo
➥$_POST['description']; ?></textarea></p>

<p><b>Kategoria 1:</b> <select name="type1">
<?php echo $pulldown; ?>
</select></p>
<p><b>Kategoria 2:</b> <select name="type2">
<?php echo $pulldown; ?>
</select></p>
<p><b>Kategoria 3:</b> <select name="type3">
<?php echo $pulldown; ?>
</select></p>
</fieldset>
<div align="center"><input type="submit"
➥name="submit" value="Wyślij" /></div>
</form>
```

Formularz (rysunek 11.9) pozwala wprowadzić do pól tekstowych kilka wartości i zapamiętać je.

11. Zakończyć skrypt PHP.

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

12. Zapisać plik pod nazwą *add_url.php*, umieścić go na serwerze WWW i przetestować przy użyciu przeglądarki internetowej (rysunek 11.10).

Rysunek 11.9. *Formularz służący do dodawania adresów URL*

Rysunek 11.10. *Po przesłaniu wprowadzonych danych zostanie wyświetlony komunikat i formularz będzie widoczny ponownie (z poprzednio wstawionymi wartościami)*

Wskazówki

- W celu zablokowania możliwości stosowania znaczników języka HTML we wprowadzanych wartościach należy posłużyć się funkcją `strip_tags()`. Funkcja ta usuwa wszystkie znaczniki nie tylko języka HTML, ale też PHP.
- Aby wyświetlić znaczniki języka HTML (w postaci, w jakiej występują w kodzie źródłowym) bez ich uaktywniania należy skorzystać z funkcji `htmlspecialchars()` i `htmlentities()`.
- Funkcja `mysql_insert_id()` jest powiązana z każdą pojedynczą sesją (interakcja z bazą danych). Z tego też powodu nie należy się przejmować tym, że zwrócono nieprawidłową wartość nawet wtedy, gdy skrypt jednocześnie został wykonany przez kilku różnych użytkowników.
- W przypadku korzystania ze skryptów PHP i stron HTML można stosować rozwiązanie zapamiętujące, jakie pozycje zostały wybrane z menu rozwijanego, ale jest to związane z większą ilością kodu źródłowego, a ponadto w przytoczonym przykładzie niezbyt dobrze współpracuje ze zmienną `$pulldown`.
- W zastosowanym przykładzie można było też stworzyć kontrolki, które posłużyłyby do wybierania maksymalnie trzech kategorii.
- Po uaktywnieniu funkcji *Magic Quotes*, wartości wprowadzone do formularza przed przesłaniem do przeglądarki muszą zostać przetworzone przez funkcję `stripslashes()`.
- W przytoczonym przykładzie po poprawnym przesłaniu wstawionych wartości formularz zostanie ponownie wyświetlony wraz z poprzednio wprowadzonymi danymi. Aby to zmienić, należy zrezygnować z wyświetlania formularza lub wyczyścić zawartość zmiennej tablicowej `$_POST` po udanym wykonaniu zapytania na bazie danych.
- Funkcja języka PHP o nazwie `mysql_insert()` jest odpowiednikiem funkcji `LAST_INSERT_ID()` oferowanej przez serwer MySQL.

Przeglądanie wysłanych adresów URL

Skrypt umożliwiający przeglądanie adresów URL będzie się składał z dwóch części — górnej i dolnej. W pierwszej z nich zostanie wyświetlone menu rozwijane z dostępnymi typami adresów, natomiast w drugiej wszystkie odnośniki dla wybranego typu. Gdy użytkownik wyświetli stronę po raz pierwszy, wtedy nie zostanie wyświetlony żaden adres URL. Po wybraniu przez niego typu i przesłaniu danych wprowadzonych do formularza, strona zostanie wyświetlona ponownie wraz z listą adresów URL dla wybranego typu (niezależnie od tego menu rozwijane w dalszym ciągu będzie widoczne).

Nowe funkcje i metody

Skrypt *view_urls.php* wprowadza dwa nowe rozwiązania. Jednym z nich jest rzutowanie, czyli narzucanie zmiennej konkretnego typu (całkowitoliczbowy, łańcuchowy itp.). Aby zrzutować zmienną, należy przed jej nazwą w nawiasach określić ostateczny jej typ. Oto przykład:

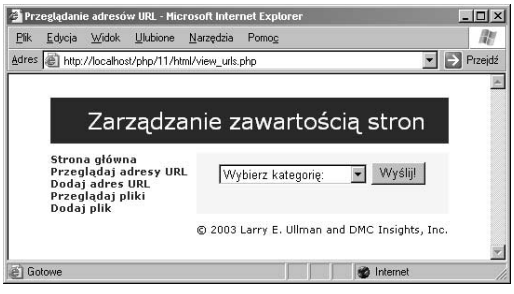
```
$var = "10"; //typ łańcuchowy
$var = (int)$var; //typ całkowitoliczbowy
```

Do dostępnych typów zmiennych (w niektórych przypadkach wiele określeń dotyczy tego samego typu) należy zaliczyć takie typy jak `int` i `integer`, `bool` i `boolean`, `float`, `double`, `real`, `string`, `array` i `object`. Interpreter języka PHP w oparciu o początkową wartość rzutowanej zmiennej zmieni jej typ i przypisze jej odpowiadającą mu wartość logiczną (w celu uzyskania szczegółowych informacji z tym związanych należy zajrzeć do dokumentacji języka PHP). Przy użyciu funkcji `intval()` można też zmienić typ zmiennej na typ całkowitoliczbowy (co zostanie pokazane w kolejnym skrypcie). Z kolei poprzez umieszczenie wartości w znakach cudzysłowu można zamienić typ zmiennej na łańcuchowy.

Drugim nowym rozwiązaniem zastosowanym w skrypcie jest funkcja `list()`. Funkcja pobiera wartości zmiennej tablicowej i przypisuje je oddzielnym zmiennym. Oto przykład:

```
$var = array ("Jan", "Nowak");
list ($first, $last) = $var;
```

Powyższym zmiennym `$first` i `$last` zostaną przypisane odpowiednio wartości *Jan* i *Nowak*. Podobna funkcjonalność może być również uzyskana przy użyciu funkcji `extract()` podobnej do funkcji `list()`.



Rysunek 11.11. *Menu rozwijane*

Aby utworzyć skrypt view_urls.php, należy wykonać poniższe kroki:

1. W edytorze tekstu utworzyć nowy skrypt PHP (listing 11.6).

```
<?php # Listing 11.6 - view_urls.php
$page_title = 'Przeglądanie adresów URL';
include_once ('includes/naglowek.html');
require_once ('../mysql_connect.php');
```

2. Rozpocząć definiowanie formularza zawartego na stronie HTML.

```
echo '<div align="center">
<form method="get" action="view_urls.php">
<select name="type">
<option value="NULL">Wybierz
➥kategorię:</option>
';
```

Po pierwszym wyświetleniu strony HTML będzie na niej widoczny formularz składający się z menu rozwijanego i przycisku *Wyślij* (rysunek 11.11). W tym miejscu zawarto początek definicji formularza umieszczonego na stronie HTML.

Listing 11.6. *Skrypt view_urls.php wyświetla zarówno menu z kategoriami adresów URL, jak i z adresami dla konkretnej kategorii*

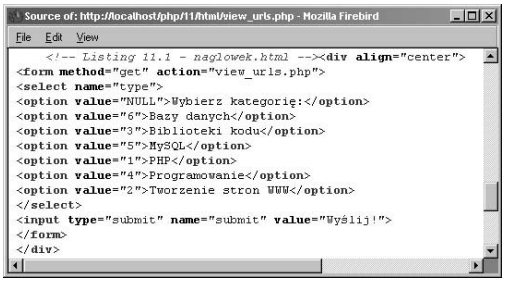
Listing

```
1   <?php # Listing 11.6 - view_urls.php
2   // Strona pozwala wyświetlić adresy URL zapisane w bazie danych.
3   // Ustawienie tytułu strony i dołączenie nagłówka HTML.
4   $page_title = 'Przeglądanie adresów URL';
5   include_once ('includes/naglowek.html');
6   require_once ('../mysql_connect.php'); // Połączenie z bazą danych.
7   // Tworzenie formularza pozwalającego użytkownikowi wybranie adresu URL,
    który zostanie wyświetlony.
8   echo '<div align="center">
9   <form method="get" action="view_urls.php">
10  <select name="type">
11  <option value="NULL">Wybierz kategorię:</option>
12  ';
```

3. Pobrać wszystkie dostępne typy adresów URL i dołączyć je do menu rozwijanego.

```
$query = 'SELECT * FROM url_types
➥ORDER BY type ASC';
$result = mysql_query ($query);
while ($row = mysql_fetch_array
➥($result, MYSQL_NUM)) {
  echo '<option value="', $row[0], '">',
  ➥stripslashes($row[1]), '</option>
';
}
```

Powyższy kod pobiera z tabeli `url_types` każdy typ adresu URL i na podstawie zwróconych rekordów generuje kod źródłowy menu rozwijanego umieszczonego na stronie HTML (rysunek 11.12).



Rysunek 11.12. *Dynamicznie wygenerowany kod źródłowy formularza zawartego na stronie HTML*

Listing 11.6. *Skrypt view_urls.php wyświetla zarówno menu z kategoriami adresów URL, jak i z adresami dla konkretnej kategorii — ciąg dalszy*

Listing

```
13  // Pobranie i wyświetlenie dostępnych typów.
14  $query = 'SELECT * FROM url_types ORDER BY type ASC';
15  $result = mysql_query ($query);
16  while ($row = mysql_fetch_array ($result, MYSQL_NUM)) {
17    echo '<option value="', $row[0], '">', stripslashes($row[1]), '</option>
18  ';
19  }
20
21  // Zakończenie formularza.
22  echo '</select>
23  <input type="submit" name="submit" value="Wyślij!">
24  </form>
25  </div>
26  ';
27
28  // Pobranie adresów URL określonego typu (jeśli go podano).
29  if (isset($_GET['type'])) {
30
31    $t = intval($_GET['type']); // Sprawdzenie, czy typ jest całkowitoliczbowy.
32
33    // Pobranie nazwy aktualnie wybranego typu.
34    $query = "SELECT type FROM url_types WHERE type_id=$t";
35    $result = mysql_query ($query);
36    list ($type) = mysql_fetch_array ($result, MYSQL_NUM);
37
38    echo "<hr /><div align=\"center\"><b>$type odnośników</b><br />
39  <small>(Wszystkie odnośniki będą korzystały z własnego okna. Najpierw
    zostaną wyświetlone ostatnio dodane odnośniki.)</small></div>\n";
```

4. Zakończyć definicję formularza umieszczonego na stronie HTML.

```
echo '</select>
<input type="submit" name="submit"
➥value="Wyślij!">
</form>
</div>
';
```

5. Sprawdzić, czy został wybrany typ adresu URL i czy pobrano informacje na jego temat.

```
if (isset($_GET['type'])) {

  $t = intval($_GET['type']);

  $query = "SELECT type FROM url_types
  ➥WHERE type_id=$t";
  $result = mysql_query ($query);
  list ($type) = mysql_fetch_array
  ➥($result, MYSQL_NUM);

  echo "<hr /><div
  ➥align=\"center\"><b>$type
  ➥odnośników</b><br />
<small>(Wszystkie odnośniki będą
➥korzystały z własnego okna. Najpierw
➥zostaną wyświetlone ostatnio dodane
➥odnośniki.)</small></div>\n";
```

Jeśli typ został już wybrany (w takim przypadku zostanie dołączony do adresu URL i udostępniony przy użyciu zmiennej tablicowej `$_GET`), powiązane z nim adresy URL powinny być pobrane.

Pierwsza operacja będzie polegała na zastosowaniu funkcji `intval()` w celu sprawdzenia, czy wybrany *typ*, który zostanie zastosowany w zapytaniu jest typem całkowitoliczbowym. W dalszej kolejności przy użyciu funkcji `list()` zostanie pobrana nazwa typu i wyświetlona na stronie w roli nagłówka. Więcej informacji na temat funkcji `intval()` i `list()` można znaleźć w ramce *Nowe funkcje i metody* lub w dokumentacji języka PHP.

6. Zainicjalizować zmienną `$first` i wysłać zapytanie do bazy danych.

```
$first = TRUE;
$query = "SELECT url, title, description
➥FROM urls AS u, url_titles AS ut WHERE
➥ut.title_id = u.title_id AND u.type_id=$t
➥AND u.approved = 'Y' ORDER BY
➥date_submitted desc";
$result = mysql_query ($query);
```

Listing 11.6. *Skrypt view_urls.php wyświetla zarówno menu z kategoriami adresów URL, jak i z adresami dla konkretnej kategorii — ciąg dalszy*

Listing

```
40    $first = TRUE; // Inicjalizacja zmiennej.
41    // Wykonanie zapytania na bazie danych.
42    $query = "SELECT url, title, description FROM urls AS u, url_titles AS ut
      WHERE ut.title_id = u.title_id AND u.type_id=$t AND u.approved = 'Y'
      ORDER BY date_submitted desc";
43    $result = mysql_query ($query);
44    // Wyświetlenie wszystkich adresów URL.
45    while ($row = mysql_fetch_array ($result, MYSQL_ASSOC)) {
46      // Jeśli jest to pierwszy rekord, wtedy zostanie utworzona tabela header.
47    if ($first) {
48      echo '<table border="0" width="100%" cellspacing="3" cellpadding="3" align="center">
49    <tr>
50      <td align="right" width="50%"><font size="+1">Odnośnik</font></td>
51      <td align="left" width="50%"><font size="+1">Opis</font></td>
52    </tr>';
53      } // Koniec pętli warunkowej ze zmienną $first.
```

Zmienna $first zostanie wykorzystana na dwa sposoby. Pierwszy z nich będzie polegał na tym, że zmienna wskaże, że przed wyświetleniem pierwszego rekordu na stronie HTML powinna zostać utworzona tabela. W drugim przypadku zmienna posłuży do sprawdzenia, czy w wyniku wykonanego zapytania zostały zwrócone jakiekolwiek adresy URL.

7. Wyświetlić wszystkie zwrócone rekordy.

```
while ($row = mysql_fetch_array
➥($result, MYSQL_ASSOC)) {

  if ($first) {
    echo '<table border="0" width="100%"
    ➥cellspacing="3" cellpadding="3"
    ➥align="center">
<tr>
  <td align="right" width="50%"><font
  ➥size="+1">Odnośnik</font></td>
  <td align="left" width="50%"><font
  ➥size="+1">Opis</font></td>
</tr>';
}
echo "  <tr>
      <td align=\"right\"><a
      ➥href=\"http://{$row['url']}\"
      ➥target=\"_new\">" . stripslashes
      ➥($row['title']) . "</a></td>
      <td align=\"left\">{$row
      ➥['description']}</td>
    </tr>\n";

    $first = FALSE;

  }
```

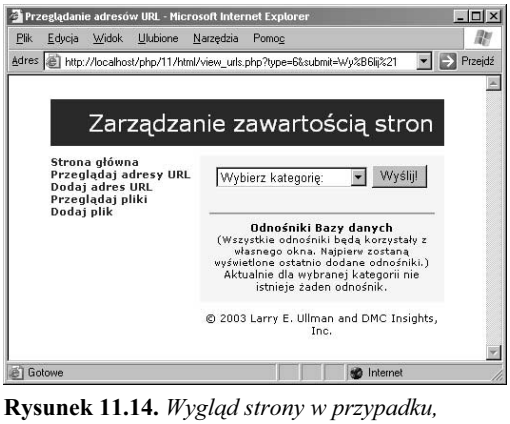
Pętla while zwróci wszystkie rekordy pobrane w wyniku wykonanego zapytania. Przed wyświetleniem pierwszego rekordu do przeglądarki internetowej zostanie wysłana tabela i jej nagłówek (rysunek11.13).

Listing 11.6. *Skrypt view_urls.php wyświetla zarówno menu z kategoriami adresów URL, jak i z adresami dla konkretnej kategorii — ciąg dalszy*

Listing

```
54      // Wyświetla wszystkie rekordy.
55      echo "  <tr>
56        <td align=\"right\"><a href=\"http://{$row['url']}\" target=\"_new\">" .
57        stripslashes($row['title']) . "</a></td>
58        <td align=\"left\">{$row['description']}</td>
59      </tr>\n";
60      $first = FALSE; // Zwrócono jeden rekord.
61    } // Koniec pętli while.
62    // Jeśli nie wyświetlono żadnego rekordu...
63    if ($first) {
64      echo '<div align="center">Aktualnie dla wybranej kategorii nie istnieje
        żaden odnośnik.</div>';
65    } else {
66      echo '</table>'; // Zamknięcie tabeli.
67    }
68  } // Koniec intstrukcji warunkowej $_GET['type'].
69  mysql_close(); // Zamknięcie połączenia z bazą danych.
70  include_once ('includes/stopka.html'); // Dołączenie stopki HTML.
71  ?>
```

Rysunek 11.13. *Lista adresów URL dla wybranego typu*



Rysunek 11.14. *Wygląd strony w przypadku, gdy z wybranym typem nie powiązano jeszcze żadnych adresów URL*

8. Wyświetlić komunikat, jeśli nie został zwrócony żaden adres URL i zakończyć kod głównej instrukcji warunkowej.

```
if ($first) {
  echo '<div align="center">Aktualnie dla
  ➥wybranej kategorii nie istnieje żaden
  ➥odnośnik.</div>';
} else {
  echo '</table>';
} }
```

Jeśli zostaną zwrócone jakiekolwiek wiersze, zmienna `$first` znajdująca się wewnątrz pętli `while` przyjmie wartość `FALSE`. A zatem, jeśli wartością zmiennej `$first` nadal będzie `TRUE`, oznacza to, że nie zwrócono żadnych rekordów i powinien zostać wyświetlony komunikat podobny do pokazanego na rysunku 11.14. W przeciwnym razie powinna zostać wygenerowana kompletna tabela.

9. Zakończyć kod źródłowy strony HTML.

```
mysql_close();
include_once ('includes/stopka.html'); ?>
```

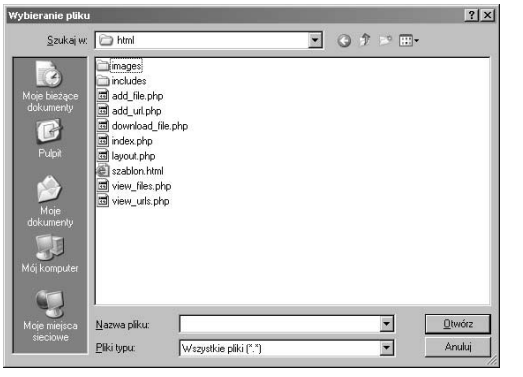
10. Zapisać plik pod nazwą *view_urls.php*, umieścić go na serwerze WWW i przetestować przy użyciu przeglądarki internetowej.

Wskazówki

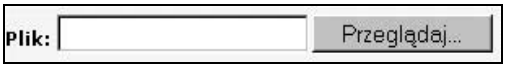
- W rozdziale 12. zamieszczono skrypt demonstrujący, w jaki sposób wyświetlać rekordy na wielu stronach (zapoznaj się z zawartością ostatniego skryptu tego rozdziału o nazwie *view_users.php*).
- Przykłady zawarte w następnym rozdziale dotyczą również operacji rejestracji i autoryzacji użytkownika. Jeśli zależy Ci na ochronie zarządzanych informacji, obie operacje powinny zostać zastosowane w aplikacji omawianej w niniejszym rozdziale.
- W razie potrzeby w stosunku do pola `description` można zastosować funkcję `nl2br()` (skrót od słów *newline to break*). Funkcja zamieni każdy znak nowego wiersza — tworzony poprzez wciśnięcie klawisza *Return* lub *Enter* — na znacznik języka (X)HTML postaci `<br/>`.

Zarządzanie plikami

Ostatnia część omawianej aplikacji internetowej służy do zarządzania plikami dowolnego typu. Odpowiednie skrypty pozwolą użytkownikom przy użyciu przeglądarki internetowej umieścić na serwerze pliki przechowywane na ich komputerach (rysunek 11.15). Dodatkowo do bazy danych zostanie dodany rekord powiązany z wykonaną operacją.



Rysunek 11.15. *Użytkownicy będą mogli wybrać plik, który zostanie umieszczony na serwerze*



Rysunek 11.16. *Definicja typu file tworzy na formularzu zawartym na stronie HTML przycisk podobny do pokazanego powyżej*

Umieszczanie plików na serwerze

Podobnie jak w przypadku przetwarzania przy użyciu skryptu PHP dowolnego formularza zawartego na stronie HTML, również operacja umieszczania plików na serwerze składa się z dwóch etapów. Najpierw musi zostać wyświetlony formularz zawarty na stronie HTML umożliwiający umieszczanie plików na serwerze. Po przesłaniu danych wprowadzonych do formularza skrypt PHP musi skopiować plik w odpowiednie miejsce na serwerze.

Wymagana składnia kodu źródłowego definiującego formularz umożliwiający umieszczenie pliku na serwerze składa się z trzech części:

```
<form enctype="multipart/form-data"
➥action="script.php" method="post">
<input type="hidden" name="MAX_FILE_SIZE"
➥value="30000">
File <input type="file" name="upload" />
```

Atrybut `enctype` zawarty w pierwszej parze znaczników kodu formularza oznacza, że powinien być w stanie obsłużyć wiele typów danych, włączając w to pliki. Należy również zauważyć, że formularz musi korzystać z metody `POST`. Wartość ukrytego pola `MAX_FILE_SIZE` spełnia w formularzu rolę ograniczenia wielkości przetwarzanego pliku (wyrażonej w bajtach) i musi znajdować się przed definicją pola `file`, które służy do utworzenia na formularzu odpowiedniego przycisku (rysunek 11.16).

Tabela 11.1. *Dane dotyczące pliku umieszczanego na serwerze mogą zostać uzyskane przy użyciu powyższych elementów tablicy*

Zmienna tablicowa $_FILES	
Indeks	**Znaczenie**
`name`	Oryginalna nazwa pliku (przechowywanego na komputerze użytkownika).
`type`	Typ MIME pliku określony przez przeglądarkę.
`size`	Wielkość pliku umieszczanego na serwerze (wyrażona w bajtach).
`tmp_name`	Tymczasowa nazwa pliku po umieszczeniu na serwerze.

Począwszy od wersji 4.1 języka PHP, dostęp do pliku umieszczonego na serwerze może być uzyskany przy użyciu zmiennej superglobalnej `$_FILES`. W przypadku wcześniejszych wersji (jeśli uaktywniono parametr `register_globals`) należało skorzystać ze zmiennej tablicowej `$HTTP_POST_FILES` lub po prostu ze zmiennej `$upload`, powiązanej z nazwą pola `file`.

Zmiennej plikowej zostaną przypisane wartości tablicy wymienione w tabeli 11.1.

Po pobraniu pliku przez skrypt PHP funkcja `move_uploaded_file()` może go przenieść z katalogu tymczasowego w jego docelowe położenie.

```
move_uploaded_file("nazwa_pliku_tymczasowego",
➥"nazwa_pliku_docelowego");
```

Po wywołaniu funkcji zakończonym powodzeniem tymczasowa wersja pliku zostanie usunięta z serwera. Aby jednak tak było, serwer WWW musi dysponować prawem zapisu do katalogu, w którym zostanie umieszczony plik.

Mając to na uwadze, na początku dokonam modyfikacji pliku konfiguracyjnego języka PHP o nazwie *php.ini* tak, aby możliwe było umieszczanie plików na serwerze, a następnie zostanie stworzony sam skrypt realizujący to zadanie.

Aby przygotować serwer, należy wykonać następujące kroki:

1. W edytorze tekstu otworzyć plik *php.ini*.
2. W *File Uploads* zmodyfikować poniższe parametry (rysunek 11.17):

```
file_uploads = On
upload_tmp_dir = /tmp
upload_max_filesize = 2M
```

   Pierwszy parametr pozwala na umieszczanie plików na serwerze lub blokuje taką możliwość. Drugi parametr pozwala określić miejsce, w którym tymczasowo będą przechowywane pliki umieszczane na serwerze. W przypadku większości systemów operacyjnych przed parametrem tym można bez większych obaw umieścić znak komentarza (znak średnika). Użytkownicy systemów Mac OS X i UNIX zazwyczaj dla parametru ustawiają katalog */tmp*, natomiast w przypadku systemu Windows powinni korzystać z katalogu określonego ścieżką *C:\PHP\uploadtemp*.

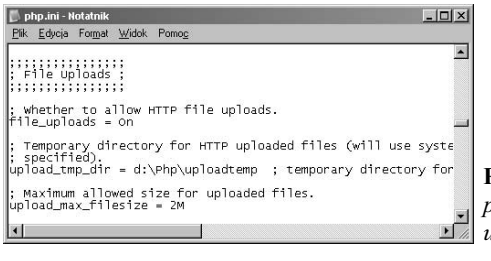
   Ostatni z wymienionych parametrów służy do określenia maksymalnej wielkości plików (wyrażonej w megabajtach) umieszczanych na serwerze.

3. Zapisać plik i ponownie uruchomić serwer WWW (rysunek 11.18).

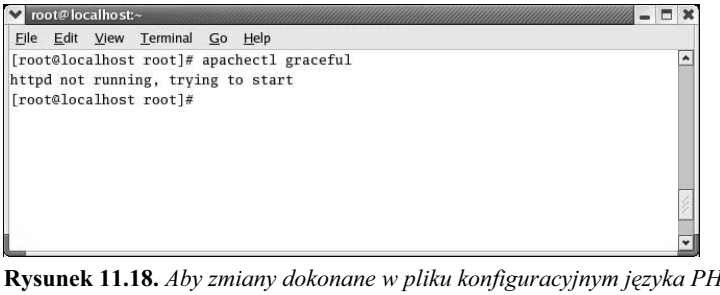
```
apachectl graceful
```

   Powyższe polecenie służy do ponownego uruchomienia serwera Apache działającego pod systemem operacyjnym UNIX. Użytkownicy systemu Mac OS X w celu ponownego uruchomienia serwera mogą użyć panelu *Sharing*, natomiast w przypadku systemu Windows należy posłużyć się apletem *Usługi* znajdującym się w *Panelu sterowania* lub innym narzędziem współpracującym z serwerem WWW.

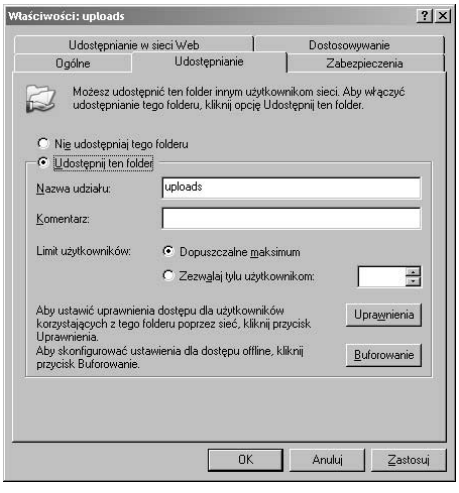
4. Utworzyć nowy katalog o nazwie *uploads* znajdujący się poza strukturą katalogową serwera WWW. Katalog ten będzie spełniał rolę docelowego miejsca, w którym będą umieszczane pliki na serwerze. Ze względu na zabezpieczenia katalog nie powinien być częścią struktury katalogowej serwera WWW.



Rysunek 11.17. *Sekcja File Uploads pliku php.ini odpowiedzialna za obsługę plików umieszczanych na serwerze*



Rysunek 11.18. *Aby zmiany dokonane w pliku konfiguracyjnym języka PHP zostały uwzględnione, konieczne jest ponowne uruchomienie serwera WWW*



Rysunek 11.19. *Karta Udostępnianie w systemie Windows pozwala na określenie, kto będzie dysponował dostępem do katalogu*

Rysunek 11.20. *Jeśli uprawnienia nie zostaną poprawnie zdefiniowane, pojawią się komunikaty błędów podobne do pokazanego*

- Ze względu na czas wymagany do umieszczenia dużego pliku na serwerze może być konieczna modyfikacja wartości parametru `max_execution_time` zawartego w pliku *php.ini* lub tymczasowe przekazanie jej do skryptu przy użyciu funkcji `set_time_limit()`.

5. Ustawić dla katalogu *uploads* takie uprawnienia, aby serwer WWW mógł w nim umieszczać pliki. Pewna, ale mniej bezpieczna metoda polega na wykonaniu polecenia `chmod 0777 uploads` (systemy UNIX i Mac OS X). W przypadku systemu Windows należy kliknąć katalog prawym przyciskiem myszy, a następnie wybrać pozycję *Udostępnianie*. Spowoduje to wyświetlenie okna zawierającego kartę *Udostępnianie* (rysunek 11.19).

W zależności od stosowanego systemu operacyjnego może się okazać, że będzie możliwe umieszczanie plików na serwerze bez konieczności wykonywania tego kroku. Aby się o tym przekonać, przed modyfikacją uprawnień należy wykonać poniższy skrypt. Jeśli zostaną wyświetlone komunikaty podobne do pokazanego na rysunku 11.20, należy dokonać zmiany w uprawnieniach.

Wskazówki

- Wartość pola `MAX_FILE_SIZE` ogranicza w przeglądarce wielkość pliku. Plik konfiguracyjny języka PHP dysponuje własnymi ograniczeniami. Można również przeprowadzić walidację wielkości pliku umieszczonego na serwerze przy użyciu skryptu PHP.
- Parametr `post_max_size` zawarty w pliku *php.ini* służy do określenia maksymalnej ilości danych (wyrażonej w megabajtach), które mogą zostać umieszczone na serwerze przy użyciu jednego skryptu. Domyślnie jest to 8 MB.
- W języku PHP w wersji 4.2 nowością jest indeks błędów zmiennej tablicowej `$_FILES`. Jego zadaniem jest przechowywanie dowolnych informacji dotyczących błędów powiązanych z plikiem umieszczanym na serwerze.
- Jeśli mamy na uwadze kwestię zabezpieczeń, przechowywanie plików umieszczanych na serwerze poza strukturą katalogów serwera WWW jest bardziej wskazane. Dzięki temu uniemożliwi się użytkownikom uzyskanie bezpośredniego dostępu do plików i uniknie się zagrożenia związanego z umieszczaniem katalogu dysponującego mało restrykcyjnymi uprawnieniami w miejscu publicznie dostępnym.

Aby utworzyć skrypt add_file.php, należy wykonać następujące kroki:

1. W edytorze tekstu utworzyć nowy skrypt PHP (listing 11.7).

```
<?php # Listing 11.7 - add_file.php
$page_title = 'Umieszczanie pliku
➥na serwerze';
include ('includes/naglowek.html');
```

2. Sprawdzić, czy wypełniono formularz i przeprowadzić walidację wartości pola `description`.

```
if (isset($_POST['submit'])) {
  require_once ('../mysql_connect.php');
  function escape_data ($data) {
    global $dbc;
    if (ini_get('magic_quotes_gpc')) {
      $data = stripslashes($data);
    }
    return mysql_real_escape_string
    ➥(trim ($data), $dbc);
  }
  if (!empty($_POST['description'])) {
    $d = escape_data
    ➥($_POST['description']);
  } else {
    $d = '';
  }
```

Podobnie jak w przypadku omawianego wcześniej skryptu *add_url.php*, dla pola `description` zostanie przeprowadzana najprostsza walidacja. Ze względu na to, że pozostałe pola formularza nie wymagają sprawdzenia, zastosowano tylko jedną instrukcję warunkową.

W celu określenia, czy rozmiar pliku umieszczanego na serwerze zawiera się w założonym przedziale można również sprawdzić jego wielkość.

Listing 11.7. *Skrypt pozwala użytkownikowi umieścić plik na serwerze*

Listing

```
1   <?php # Listing 11.7 - add_file.php
2   // Strona pozwala użytkownikom umieścić pliki na serwerze.
3   // Ustawienie tytułu strony i dołączenie nagłówka HTML.
4   $page_title = 'Umieszczanie pliku na serwerze';
5   include ('includes/naglowek.html');
6   if (isset($_POST['submit'])) { // Obsługa formularza.
7     require_once ('../mysql_connect.php'); // Połączenie z bazą danych.
8     // Funkcja usuwająca znak '/' i obcinająca dane wprowadzone w formularzu.
9     function escape_data ($data) {
10      global $dbc;
11      if (ini_get('magic_quotes_gpc')) {
12        $data = stripslashes($data);
13      }
14      return mysql_real_escape_string (trim ($data), $dbc);
15    } // Koniec sekcji funkcji escape_data().
```

3. Wstawić do bazy danych rekord powiązany z plikiem umieszczanym na serwerze.

```
$query = "INSERT INTO uploads (file_name,
➥file_size, file_type, description,
➥upload_date) VALUES
➥('{$_FILES['upload']['name']}',
➥{$_FILES['upload']['size']},
➥'{$_FILES['upload']['type']}',
➥'$d', NOW())";
$result = @mysql_query ($query);
```

Informacja o każdym pliku umieszczonym na serwerze zostanie zapisana w bazie danych. W tym celu zostanie zastosowana wielowymiarowa zmienna tablicowa `$_FILES`, która będzie przechowywała oryginalną nazwę pliku, jego wielkość i typ MIME. Wszystkie te dane zostaną pobrane z przeglądarki internetowej. Dodatkowo zostanie zapisany opis oraz aktualna data i godzina.

4. Utworzyć nowy plik.

```
if ($result) {
  $extension = explode ('.',
  ➥$_FILES['upload']['name']);
  $uid = mysql_insert_id();
  $filename = $uid . '.' . $extension[1];
```

Plik zostanie zapisany na serwerze pod nową nazwą. Takie rozwiązanie jest o wiele bardziej bezpieczne niż w przypadku posługiwania się oryginalną nazwą nadaną przez użytkownika. Nazwa pliku będzie złożona z wartości bazodanowego pola `upload_id` (pobranej przy użyciu funkcji `mysql_insert_id()`), za którą zostanie wstawiona kropka i rozszerzenie oryginalnego pliku (uzyskanego poprzez rozkład jego nazwy). Przykładowo, dokument o nazwie *chapter.doc* może przyjąć nazwę *231.doc*, natomiast plik *imagename.jpg* zostanie zapisany jako *49.jpg*.

Listing 11.7. *Skrypt pozwala użytkownikowi umieścić plik na serwerze — ciąg dalszy*

Listing

```
16   // Sprawdzenie obecności opisu (nie jest wymagane).
17   if (!empty($_POST['description'])) {
18     $d = escape_data($_POST['description']);
19   } else {
20     $d = '';
21   }
22
23   // Dodanie rekordu do bazy danych.
24   $query = "INSERT INTO uploads (file_name, file_size, file_type,
     description, upload_date) VALUES ('{$_FILES['upload']['name']}',
     {$_FILES['upload']['size']}, '{$_FILES['upload']['type']}', '$d', NOW())";
25   $result = @mysql_query ($query);
26
27   if ($result) {
28
29     // Utworzenie nazwy pliku.
30     $extension = explode ('.', $_FILES['upload']['name']);
31     $uid = mysql_insert_id(); // Identyfikator pliku umieszczanego na
       serwerze
32     $filename = $uid . '.' . $extension[1];
33
34     // Przenoszenie pliku.
35     if (move_uploaded_file($_FILES['upload']['tmp_name'],
       "../uploads/$filename")) {
36       echo '<p>Plik został umieszczony na serwerze!</p>';
```

5. Skopiować plik w docelowe miejsce na serwerze.

```
if (move_uploaded_file($_FILES
➥['upload']['tmp_name'], "../uploads/
➥$filename")) {
  echo '<p>Plik został umieszczony
  ➥na serwerze!</p>';
} else {
  echo '<p><font color="red">
  ➥Przeniesienie pliku nie było
  ➥możliwe.</font></p>';
  $query = "DELETE FROM uploads WHERE
  ➥upload_id = $uid";
  $result = @mysql_query ($query);
}
```

W celu przeniesienia pliku tymczasowego w docelowe miejsce (katalog *uploads*, gdzie plik uzyska nową nazwę) należy użyć funkcji `move_uploaded_file()`. Jeśli przeniesienie pliku nie będzie możliwe (rysunek 11.20), z bazy danych zostanie usunięty rekord i pojawi się komunikat błędu.

6. Dokończyć tworzenie instrukcji warunkowej i skryptu PHP.

```
  } else {
    echo '<p><font color="red">Na skutek
    ➥błędu systemowego wysłane zapytanie
    ➥nie mogło zostać przetworzone.
    ➥Przepraszamy za zaistniałą
    ➥niedogodność.</font></p>';
  }
  mysql_close();
}
?>
```

Listing 11.7. *Skrypt pozwala użytkownikowi umieścić plik na serwerze — ciąg dalszy*

Listing

```
37      } else {
38        echo '<p><font color="red">Przeniesienie pliku nie było możliwe.</font></p>';
39
40        // Usunięcie rekordu z bazy danych.
41        $query = "DELETE FROM uploads WHERE upload_id = $uid";
42        $result = @mysql_query ($query);
43      }
44
45    } else { // Jeśli zapytanie na bazie danych nie zostanie wykonane prawidłowo.
46      echo '<p><font color="red">Na skutek błędu systemowego wysłane zapytanie
        nie mogło zostać przetworzone. Przepraszamy za zaistniałą niedogodność.</font></p>';
47    }
48
49  mysql_close(); // Zamknięcie połączenia z bazą danych.
50
51  } // Koniec instrukcji warunkowej obsługującej dane wprowadzane w formularzu.
52  ?>
53
54  <form enctype="multipart/form-data" action="<?php echo $_SERVER['PHP_SELF']; ?>"
    method="post">
55
56  <input type="hidden" name="MAX_FILE_SIZE" value="524288">
```

7. Zdefiniować formularz zawarty na stronie HTML.

```
<form enctype="multipart/form-data"
➥action="<?php echo $_SERVER['PHP_SELF'];
➥?>" method="post">

<input type="hidden" name="MAX_FILE_SIZE"
➥value="524288">

<fieldset><legend>W celu umieszczenia
➥pliku na serwerze należy wypełnić
➥formularz:</legend>

<p><b>Plik:</b> <input type="file"
➥name="upload" /></p>

<p><b>Opis:</b> <textarea
➥name="description" cols="40"
➥rows="5"></textarea></p>

</fieldset>

<div align="center"><input type="submit"
➥name="submit" value="Wyślij" /></div>

</form>
```

Co prawda formularz jest bardzo prosty, ale zawiera trzy niezbędne elementy związane z umieszczaniem plików na serwerze — atrybut `enctype`, ukryte pole `MAX_FILE_SIZE` i pole `file`.

8. Zakończyć tworzenie skryptu PHP.

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

Listing 11.7. *Skrypt pozwala użytkownikowi umieścić plik na serwerze — ciąg dalszy*

Listing

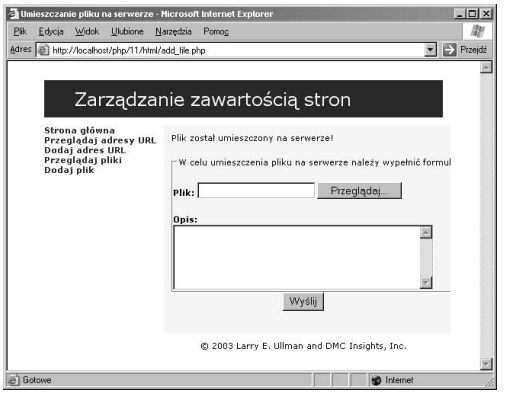
```
57  <fieldset><legend>W celu umieszczenia pliku na serwerze należy wypełnić formularz:</legend>
58
59  <p><b>Plik:</b> <input type="file" name="upload" /></p>
60
61  <p><b>Opis:</b> <textarea name="description" cols="40" rows="5"></textarea></p>
62
63  </fieldset>
64
65  <div align="center"><input type="submit" name="submit" value="Wyślij" /></div>
66
67  </form><!-- Koniec formularza-->
68
69  <?php
70  include ('includes/stopka.html'); // Dołączenie stopki HTML.
71  ?>
```

9. Zapisać plik pod nazwą *add_file.php*, umieścić go na serwerze WWW i przetestować przy użyciu przeglądarki internetowej (rysunki 11.21 i 11.22).

Wskazówki

- Obecność pliku umieszczonego na serwerze może również zostać sprawdzona przy użyciu funkcji `is_uploaded_file()`.
- W celu odwołania się do katalogów użytkownicy systemu Windows muszą skorzystać ze znaku `/` lub `\\`, a zatem zamiast ścieżki *C:* należy użyć *C:/* lub *C:*. Wynika to stąd, że w języku PHP znak `\` jest wykorzystywany przez funkcję `escape_data()`.
- Jeśli jest wykorzystywana starsza wersja języka PHP, może nie być możliwe zastosowanie zmiennej tablicowej `$_FILES`. Na skutek tego zamiast funkcji `move_uploaded_file()` konieczne będzie użycie funkcji `copy()`.
- Jeśli oba pliki mają taką samą nazwę, funkcja `move_uploaded_file()` nadpisze istniejący plik bez wcześniejszego ostrzeżenia.
- Po przygotowaniu ostatecznej wersji produkcyjnej skryptu w celu wyeliminowania w przyszłości wszelkich komunikatów błędów należy przed wywołaniami funkcji `move_uploaded_file()` wstawić znak `@`.

Rysunek 11.21. *Formularz służący do umieszczania plików na serwerze*



Rysunek 11.22. *Wygląd strony po udanym umieszczeniu pliku na serwerze*

Moduł administracyjny aplikacji

Chociaż w przypadku omawianej aplikacji nie stworzono jej części administracyjnej, to jednak nie jest to zadanie trudne do zrealizowania. Na początku należy do tabeli `uploads` dodać pole `approved`. Następnie należy tak zmodyfikować zarówno plik *add_url.php*, jak i *add_file.php*, aby domyślnie wartość pola `approved` wynosiła *N*. Z myślą o części administracyjnej aplikacji należy stworzyć stronę, na której zostaną wyświetlone wszystkie elementy, które jeszcze nie zatwierdzono. W celu zaakceptowania dowolnego elementu należy dla odpowiedniego rekordu tabeli wykonać polecenie `UPDATE` i zmienić wartość pola `approved` na *Y*.

Jedną z zalet systemu służącego do umieszczania plików na serwerze jest łatwość kojarzenia plików z danymi na ich temat przechowywanymi w bazie danych (gdy do tworzenia nazw plików jest wykorzystywana wartość pola `upload_id`). Jeśli zostanie utworzony skrypt administracyjny służący do usuwania plików z serwera (z katalogu *uploads*), można również usunąć z tabeli powiązany z nim rekord.

Przeglądanie i pobieranie plików

Ostatnie dwa skrypty omawianej aplikacji pozwolą użytkownikom przeglądnąć pliki umieszczone na serwerze, a następnie je pobrać (w danej chwili tylko jeden). Skrypt służący do przeglądania plików jest dość prosty, natomiast skrypt umożliwiający ich pobieranie wymaga częstego stosowania funkcji języka PHP o nazwie `header()`.

W celu stworzenia skryptu *view_files.php* należy wykonać następujące kroki:

1. W edytorze tekstu utworzyć nowy skrypt PHP (listing 11.8).

```
<?php # Listing 11.8 - view_files.php
$page_title = 'Przeglądanie plików';
include_once ('includes/naglowek.html');

require_once ('../mysql_connect.php');

$first = TRUE;
```

   Zmienna `$first` spełnia taką samą rolę jak w skrypcie *view_urls.php*. Zmienna zostanie wykorzystana przy tworzeniu nagłówka i w celu określenia, czy będzie możliwe przeglądanie niektórych plików.

2. Pobrać z bazy danych informacje o wszystkich plikach.

```
$query = "SELECT upload_id, file_name,
➥ROUND(file_size/1024) AS fs, description,
➥DATE_FORMAT(upload_date, '%M %e, %Y') AS d
➥FROM uploads ORDER BY upload_date DESC";
$result = mysql_query ($query);
```

   Powyższe zapytanie zwróci dla każdego pliku umieszczonego na serwerze wartości przechowywane w polach `upload_id`, `file_name`, `description`, a ponadto sformatowaną datę. Na początku zostaną wyświetlone informacje dotyczące najnowszych plików. Jednocześnie zapytanie pozwoli na uzyskanie wielkości plików (wyrażonej w kilobajtach), co będzie możliwe poprzez podzielenie przechowywanej wartości przez 1024, a następnie zaokrąglenie otrzymanej liczby.

Listing 11.8. *Skrypt view_files.php wyświetla pliki umieszczone na serwerze wraz z ich opisem, wielkością i datą wykonania operacji*

Listing

```
<?php # Listing 11.8 - view_files.php
// Strona pozwala użytkownikom przeglądać pliki umieszczone na serwerze.
// Ustawienie tytułu strony i dołączenie nagłówka HTML.
$page_title = 'Przeglądanie plików';
include_once ('includes/naglowek.html');

require_once ('../mysql_connect.php'); // Połączenie z bazą danych.

$first = TRUE; // Inicjalizacja zmiennej.
// Wysłanie zapytania do bazy danych.
$query = "SELECT upload_id, file_name, ROUND(file_size/1024) AS fs,
description, DATE_FORMAT(upload_date, '%M %e, %Y') AS d FROM uploads ORDER BY upload_date DESC";
$result = mysql_query ($query);
// Wyświetlenie wszystkich adresów URL.
while ($row = mysql_fetch_array ($result, MYSQL_ASSOC)) {
  // Jeśli jest to pierwszy rekord, zostanie utworzona tabela header.
  if ($first) {
    echo '<table border="0" width="100%" cellspacing="3" cellpadding="3" align="center">
  <tr>
    <td align="left" width="20%"><font size="+1">Nazwa pliku</font></td>
    <td align="left" width="40%"><font size="+1">Opis</font></td>
    <td align="center" width="20%"><font size="+1">Rozmiar pliku</font></td>
    <td align="left" width="20%"><font size="+1">Data umieszczenia pliku na
    serwerze</font></td>
  </tr>';
  } // Koniec pętli warunkowej ze zmienną $first.
  // Wyświetla wszystkie rekordy.
  echo "  <tr>
    <td align=\"left\"><a href= \"download_file.php?uid={$row['upload_id']}
    \">{$row ['file_name']}</a></td>
    <td align=\"left\">" . stripslashes($row['description']) . "</td>
    <td align=\"center\">{$row['fs']}kb</td>
    <td align=\"left\">{$row['d']}</td>
  </tr>\n";
  $first = FALSE; // Zwrócono jeden rekord.
} // Koniec pętli while.
// Jeśli nie wyświetlono żadnego rekordu...
if ($first) {
  echo '<div align="center">Aktualnie nie ma żadnego pliku.</div>';
} else {
  echo '</table>'; // Zamknięcie tabeli.
}
mysql_close(); // Zamknięcie połączenia z bazą danych.
include_once ('includes/stopka.html'); // Dołączenie stopki HTML.
?>
```

3. Wyświetlić wszystkie rekordy.

```
while ($row = mysql_fetch_array ($result,
➥MYSQL_ASSOC)) {
  if ($first) {
    echo '<table border="0" width="100%"
    ➥cellspacing="3" cellpadding="3"
    ➥align="center">
  <tr>
    <td align="left" width="20%"><font
    ➥size="+1">Nazwa pliku</font></td>
    <td align="left" width="40%"><font
    ➥size="+1">Opis</font></td>
    <td align="center" width="20%"><font
    ➥size="+1">Rozmiar pliku</font></td>
    <td align="left" width="20%"><font
    ➥size="+1">Data umieszczenia pliku
    ➥na serwerze</font></td>
  </tr>';
  }
  echo "  <tr>
    <td align=\"left\"><a href=
    ➥\"download_file.php?uid=
    ➥{$row['upload_id']}\">{$row
    ➥['file_name']}</a></td>
    <td align=\"left\">" .
    ➥stripslashes($row['description']) .
    ➥"</td>
    <td align=\"center\">
    ➥{$row['fs']}kb</td>
    <td align=\"left\">{$row['d']}</td>
  </tr>\n";
  $first = FALSE;
}
```

Również w tym przypadku powyższy kod działa podobnie jak w poprzednim skrypcie *view_urls.php*. Zmienna `$first` jest używana raz przy tworzeniu nagłówka (rysunek 11.23), po czym są wyświetlane wszystkie rekordy. Przy odwoływaniu się do funkcji `mysql_fetch_array()` jest stosowana stała `MYSQL_ASSOC`, np. w celu wyświetlenia wartości należy użyć instrukcji `$row['fs']`.

Każda nazwa pliku spełnia rolę odnośnika do skryptu *download_file.php*, przy czym do adresu URL jest dołączana wartość pola `upload_id`. Wartość ta zostanie wykorzystana przez skrypt służący do pobierania plików w celu określenia, który z nich ma być przesłany do przeglądarki internetowej.

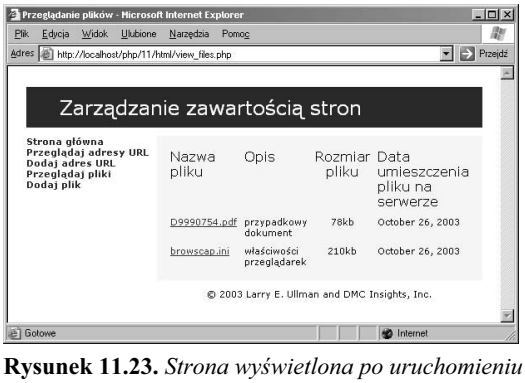
4. Jeśli nie znaleziono żadnych plików lub zamknięto tabelę, należy wyświetlić komunikat.

```
if ($first) {
  echo '<div align="center">Aktualnie
  ➥nie ma żadnego pliku.</div>';
} else {
  echo '</table>';
}
```

5. Zakończyć tworzenie skryptu PHP.

```
mysql_close();
include_once ('includes/stopka.html');  ?>
```

6. Zapisać plik pod nazwą *view_files.php*, umieścić go na serwerze WWW i przetestować przy użyciu przeglądarki internetowej.



Rysunek 11.23. *Strona wyświetlona po uruchomieniu skryptu view_files.php*

Aby utworzyć skrypt download_file.php, należy wykonać następujące kroki:

1. W edytorze tekstu utworzyć nowy skrypt PHP (listing 11.9).

```
<?php # Listing 11.9 - download_file.php
```

2. Sprawdzić wartość pola `upload_id`.

```
if (is_numeric ($_GET['uid'])) {
```

Przed kontynuacją należy upewnić się, czy skrypt pobrał poprawną wartość pola `upload_id`, która powinna być liczbą.

3. Pobrać informacje na temat pliku.

```
require_once ('../mysql_connect.php');

$query = "SELECT file_name, file_type,
➥file_size FROM uploads WHERE upload_id
➥= {$_GET['uid']}";
$result = mysql_query ($query);
list ($fn, $ft, $fs) = mysql_fetch_array
➥($result, MYSQL_NUM);
mysql_close();
```

W celu pobrania pliku konieczna jest znajomość jego nazwy, typu, wielkości. Wszystkie te dane zostaną pobrane z bazy danych poprzez wykonanie zapytania, w którym umieszczono instrukcje `$_GET['uid']`, a następnie wywołanie funkcji `list()`.

4. Określić nazwę pliku.

```
$extension = explode ('.', $fn);
$the_file = '../uploads/' . $_GET['uid']
➥. '.' . $extension[1];
```

Działanie powyższego kodu jest podobne do funkcjonowania mechanizmu nazewniczego zastosowanego w skrypcie *add_file.php*, z tą różnicą, że w tym przypadku dodano ścieżkę postaci *../uploads/*. Przypisanie takiego łańcucha zmiennej ułatwi odwoływanie się do pliku w dalszej części skryptu.

Listing 11.9. *Poprzez przesłanie do przeglądarki internetowej odpowiednich nagłówków skrypt wymusza wykonanie operacji pobrania pliku*

Listing

```
1   <?php # Listing 11.9 - download_file.php
2   // Strona pozwala na pobranie pliku z serwera przy użyciu funkcji header().
3
4   if (is_numeric ($_GET['uid'])) { // Identyfikator pliku umieszczonego na serwerze.
5
6     require_once ('../mysql_connect.php'); // Połączenie z bazą danych.
7
8     // Pobranie informacji na temat pliku.
9     $query = "SELECT file_name, file_type, file_size FROM uploads WHERE upload_id = {$_GET['uid']}";
10    $result = mysql_query ($query);
11    list ($fn, $ft, $fs) = mysql_fetch_array ($result, MYSQL_NUM);
12    mysql_close(); // Zamknięcie połączenia z bazą danych.
13    // Określenie nazwy pliku umieszczonego na serwerze.
14    $extension = explode ('.', $fn);
15    $the_file = '../uploads/' . $_GET['uid'] . '.' . $extension[1];
16
17    // Sprawdzenie, czy plik istnieje.
18    if (file_exists ($the_file)) {
```

5. Sprawdzić, czy plik istnieje na serwerze.

```
if (file_exists ($the_file)) {
```

Przed wysłaniem pliku do przeglądarki internetowej należy upewnić się, czy istnieje. Jeśli plik istnieje na serwerze, funkcja `file_exists()` zwróci wartość *TRUE*.

6. Wysłać plik.

```
header ("Content-Type: application/$ft");
header ("Content-disposition: attachment;
➥filename=$fn");
header ("Content-Length: $fs");
readfile ($the_file);
$message = '<p>Plik został wysłany.</p>';
```

Wywołania funkcji `header()` powodują przesłanie do przeglądarki pliku i utworzenie okna dialogowego (rysunek 11.24). W oparciu o typ MIME (zapisany w bazie danych po umieszczeniu pliku na serwerze) pierwszy wiersz ma za zadanie przygotowanie przeglądarki do odebrania pliku. Drugi wiersz przy użyciu oryginalnej nazwy pliku przechowywanego na komputerze użytkownika określa nazwę pobieranego pliku. Ostatnia funkcja `header()` informuje o ilości przesyłanych danych. Również ta informacja została ustalona po umieszczeniu pliku na serwerze. Zawartość pliku jest przesyłana przy użyciu funkcji `readfile()`, która po jego odczytaniu natychmiast przekazuje zawartość strony do przeglądarki internetowej.

Listing 11.9. *Poprzez przesłanie do przeglądarki internetowej odpowiednich nagłówków skrypt wymusza wykonanie operacji pobrania pliku — ciąg dalszy*

Listing

```
19      // Wysłanie pliku.
20      header ("Content-Type: application/$ft");
21      header ("Content-disposition: attachment; filename=$fn");
22      header ("Content-Length: $fs");
23      readfile ($the_file);
24
25      $message = '<p>Plik został wysłany.</p>';
26
27    } else { // Plik nie istnieje.
28      $message = '<p><font color="red">Odnalezienie pliku na serwerze nie było
        możliwe. Przepraszamy za zaistniałą niedogodność.</font></p>';
29    }
30
31  } else { // Nieprawidłowy identyfikator pliku umieszczonego na serwerze.
32
33    $message = '<p><font color="red">Proszę wybrać poprawny plik znajdujący się
      na serwerze.</font></p>';
34
35  }
36
37  // Ustawienie tytułu strony i dołączenie nagłówka HTML.
38  $page_title = 'Pobieranie pliku';
39  include_once ('includes/naglowek.html');
40
41  echo $message;
42
43  include_once ('includes/stopka.html');
44  ?>
```

7. Zakończyć instrukcje warunkowe.

```
    } else {
      $message = '<p><font color=
      ➥"red">Odnalezienie pliku na serwerze
      ➥nie było możliwe. Przepraszamy za
      ➥zaistniałą niedogodność.</font></p>';
    }
} else {
    $message = '<p><font color="red">Proszę
    ➥wybrać poprawny plik znajdujący się
    ➥na serwerze.</font></p>';
}
```

8. Wprowadzić zawartość strony internetowej.

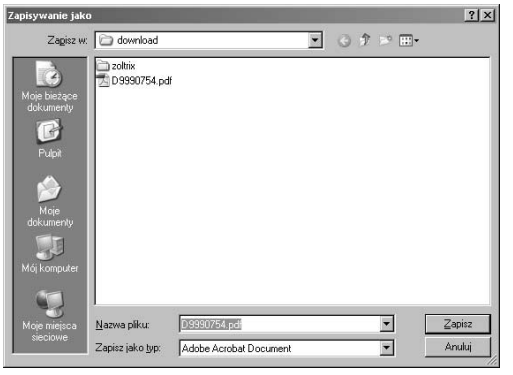
```
$page_title = 'Pobieranie pliku';
include_once ('includes/naglowek.html');
echo $message;
include_once ('includes/stopka.html');
?>
```

Tak naprawdę większość przeglądarek internetowych nigdy nie wyświetli tych informacji. Większość użytkowników po kliknięciu odnośnika wywołującego skrypt *view_files.php* zobaczy jedynie okno dialogowe umożliwiające pobranie pliku. Wykorzystanie powyższego kodu źródłowego nie spowoduje obniżenia stopnia zabezpieczeń. Dodatkowo kod zostanie zastosowany w przypadku wystąpienia jakiegokolwiek problemu.

9. Zapisać plik pod nazwą *download_file.php*, umieścić go na serwerze WWW i przetestować przy użyciu przeglądarki internetowej.

Wskazówki

- W języku PHP w wersji 4.3 nowością jest funkcja `mime_content_type()` zwracająca typ MIME dla pliku znajdującego się na serwerze.
- Przeglądarki internetowe w różny sposób obsługują operację pobierania plików. Z tego też powodu w celu sprawdzenia zgodności uzyskiwanych wyników skrypt podobny do powyższego powinien zostać przetestowany w miarę możliwości na jak największej liczbie przeglądarek i platform.



Rysunek 11.24. *Kliknięcie odnośnika wywołującego skrypt view_files.php powinno spowodować wyświetlenie pokazanego okna dialogowego*