Tematy: Przygotowanie formularza | Przesyłanie danych przez formularz | Mechanizm PostBack

W rozdziale 1. pojawiła się prosta definicja strony WWW — definicja ta określiła dynamiczną stronę WWW jako stronę, która jest generowana na podstawie danych przekazanych m.in. przez użytkownika. Jest to oczywiście prawda, dlatego też w rozdziale tym poruszony zostanie temat przetwarzania danych dostarczanych bezpośrednio przez użytkownika.

Podstawowym sposobem interakcji między użytkownikiem a serwisem WWW są formularze HTML. Formularze HTML to specjalna konstrukcja języka HTML, dzięki której przeglądarka generuje zestaw elementów kontrolnych (znanych z „okienkowych" interfejsów użytkownika), pozwalających użytkownikowi na wprowadzanie danych i podejmowanie działań typu wybór elementu z listy rozwijanej, kliknięcie przycisku itp. Na pewno każdy użytkownik internetu spotkał się podczas surfowania z różnymi rodzajami formularzy — od najprostszych, stosowanych w wyszukiwarkach internetowych (jedno pole do wprowadzenia wyszukiwanej frazy i jeden przycisk, zwykle o nazwie Szukaj), do bardziej skomplikowanych, stosowanych na stronach wymagających rejestracji, gdzie należy podać wiele danych o sobie itp.

W rozdziale tym zamieszczone zostały informacje na temat tworzenia formularzy HTML, obsługi danych z formularzy za pomocą protokołu HTTP i przetwarzania danych z formularzy w PHP.

Przygotowanie formularza

Zanim zagłębimy się w kod PHP, spróbujemy utworzyć formularz HTML. Na potrzeby tego rozdziału stworzymy sobie prosty formularz rejestracji, za pomocą którego użytkownik będzie mógł przesłać dane rejestracyjne (imię, nazwisko, zawód, adres e-mail i zgodę na dostarczanie informacji handlowych) do serwisu.

Na początku utworzymy formularz w czystym języku HTML i przetestujemy go za pomocą przeglądarki WWW.

Dokładny opis znaczników (X)HTML, za pomocą których tworzone są formularze, znaleźć można w specyfikacji języka (X)HTML lub w jednym z wielu dobrych kursów tego języka. Ponieważ język (X)HTML nie jest tematem tego kursu, znaczniki (X)HTML nie będą tu szczegółowo opisywane — wyjątek stanowić będą tzw. konieczne przypadki, czyli te, które takiego wyjaśnienia będą wymagały.

Ogólna postać formularza jest następująca:

```
<form  name = &quot;nazwa&quot;

    target = &quot;okno&quot;

    action = &quot;url&quot;

    method = &quot;metoda&quot;

    enctype = &quot;typ kodowania&quot;>

<!-- tu  definicja obiektów składowych-->

</form>
```

Parametr method wskazuje metodę, która zostanie użyta do przesłania danych do serwera, najczęściej jest to GET lub POST, natomiast action określa adres skryptu, który będzie odbierał dane. Może on być adresem

bezwzględnym, np. http://www.mojadomena.com/skrypt.php, lub względnym, np. /skrypty/skrypt.php. W pierwszym przypadku dane zostaną dostarczone do skryptu skrypt.php znajdującego się w głównym katalogu serwera o adresie http://www.mojadomena.com, natomiast w drugim — do skryptu o nazwie skrypt.php znajdującego się w podkatalogu skrypty na serwerze ze stroną WWW zawierającą formularz.

Elementami składowymi formularza mogą być:

- button — klasyczny przycisk,
- checkbox — pole wyboru,
- hidden — element ukryty,
- password — pole tekstowe do wpisywania haseł,
- radio — pole wyboru,
- reset — przycisk reset,
- select — lista wyboru,
- submit — przycisk submit,
- text — pole tekstowe,
- textarea — rozszerzone pole tekstowe.

Na listingu 5.1 widoczny jest kod HTML przykładowego formularza.

Listing 5.1. Kod HTML strony WWW zawierającej przykładowy formularz

```
<!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>Formularz</title>
</head>
```

```
<body>

  <form action=&quot;&quot; method=&quot;post&quot;>

  <div>

  Imię:<br />

  <input name=&quot;imie&quot; value=&quot;&quot;  /><br />

  Nazwisko:<br />

  <input name=&quot;nazwisko&quot; value=&quot;&quot;  /><br />

  Zawód:<br />

  <input name=&quot;zawod&quot; value=&quot;&quot;  /><br />

  Adres e-mail:<br />

  <input name=&quot;email&quot; value=&quot;&quot;  /><br />

  <input type=&quot;checkbox&quot;  value=&quot;checked&quot; name=&quot;mailing&quot; />Chcę otrzymywać
informacje handlowe<br /><br />

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

  </div>

  </form>

</body>

</html>
```

Po wyświetleniu w przeglądarce WWW formularz będzie wyglądał podobnie, jak przedstawiony na rysunku 5.1.

Rysunek 5.1. Przykładowy formularz

Nasz formularz składa się z czterech pól do wprowadzania tekstu (zdefiniowanych za pomocą znaczników <input> z atrybutem type ustawionym na text), jednego pola wyboru (zdefiniowanego za pomocą znacznika <input> z atrybutem type ustawionym na checkbox) oraz jednego przycisku (znacznik <input> z atrybutem type ustawionym na submit). Każdy z wymienionych elementów ma atrybut name, któremu przypisana jest nazwa identyfikująca jednoznacznie ten element formularza. Nagłówek formularza (znacznik form) zawiera dwa atrybuty: action i method — atrybuty te zostaną w miarę szczegółowo opisane w dalszej części rozdziału. Ostatnim elementem formularza jest przycisk, za pomocą którego dane z formularza zostaną przesłane dalej.

Przesyłanie danych przez formularz

Nietrudno się domyślić, że po kliknięciu przycisku Wyślij (rysunek 5.1) dane z formularza przesyłane są dalej za

pomocą protokołu HTTP — tak jak reszta elementów serwisu. O tym, w jaki sposób dane z formularza są przesyłane dalej, decyduje metoda zdefiniowana w atrybucie method znacznika form — w przypadku przykładowego formularza jest to metoda POST protokołu HTTP.

Oprócz metody POST można również użyć metody GET, ale musimy pamiętać o jej ograniczeniach: przede wszystkim o tym, że dane z formularza przesyłane są za pomocą adresu URL, czyli w sposób widoczny dla użytkownika (zwykły użytkownik nawet nie zwróci na to uwagi, ale są użytkownicy, dla których informacje zawarte w tak rozbudowanym adresie mogą być bardzo przydatne…). URL będzie miał wtedy schematyczną postać:

http://adres.serwera/skrypt.php?parametr1=wartość1&parametr2=wartość2

Dodatkowym ograniczeniem jest długość adresu — maksymalnie 255 znaków. W przypadku zastosowania metody POST dane przesyłane przez formularz nie są widoczne dla użytkownika — aby je zobaczyć, należałoby użyć dowolnego programu do podglądu danych przesyłanych przez sieć — i nie mają takiego ograniczenia co do długości, jak w przypadku zastosowania metody GET.

A jak jest z formularzami w PHP? Chyba najlepszym sposobem wyjaśnienia obsługi formularzy w języku PHP będzie analiza przykładu. Listing 5.2 zawiera kod PHP strony generującej przykładowy formularz (jest to modyfikacja kodu HTML z listingu 5.1).

Listing 5.2. Strona (skrypt PHP) generująca formularz

```
<!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>Formularz</title>

</head>

<?php

 // Funkcja wyświetlająca formularz

 function formularz() {

  ?>

  <form action=&quot;zgloszenie.php&quot; method=&quot;post&quot;>

  <div>

  Imię:<br />

  <input name=&quot;imie&quot; value=&quot;&quot; /><br />

  Nazwisko:<br />

  <input name=&quot;nazwisko&quot; value=&quot;&quot; /><br />
```

```
    Zawód:<br />

    <input name=&quot;zawod&quot; value=&quot;&quot;  /><br />

    Adres e-mail:<br />

    <input name=&quot;email&quot; value=&quot;&quot;  /><br />

    <input type=&quot;checkbox&quot;  name=&quot;mailing&quot; value=&quot;checked&quot; />Chcę
otrzymywać  informacje handlowe<br /><br />

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

    </div>

    </form>

    <?php

  }

?>

<body>

<?php

  formularz();

?>

</body>

</html>
```

Kod z listingu 5.2 na pierwszy rzut oka wydaje się skomplikowany — nie bez racji, ponieważ jest napisany trochę na wyrost (będzie dalej modyfikowany).

Zacznijmy jednak od początku. Pierwsza sekcja kodu PHP zawiera definicję funkcji formularz(), której jedynym zadaniem jest wyświetlanie formularza. Funkcja ta nie zwraca żadnych wartości, dlatego nie zawiera słowa kluczowego return. Można jednak w ciele tej funkcji znaleźć pewną ciekawą rzecz: zaraz po nawiasie otwierającym blok ciała funkcji zamykana jest sekcja kodu PHP i dalej widać już tylko kod HTML definiujący formularz. Po znaczniku zamykającym formularz pojawia się znów symbol otwarcia sekcji PHP, po czym zamykane jest ciało funkcji i zamykana jest również sekcja kodu PHP. Mechanizm zastosowany w funkcji formularz() nazywany jest wyskakiwaniem z PHP (opuszczaniem trybu PHP) — po prostu kod HTML znajdujący się wewnątrz funkcji zostanie wyświetlony w oknie przeglądarki po wywołaniu tej funkcji. Można by oczywiście nie stosować wyskakiwania, ale wtedy musielibyśmy wypisać kod HTML za pomocą jednej (lub kilku) instrukcji echo lub print. Oczywiście czasem stosowanie wyskakiwania może się nie opłacać (ze względu na nakład pracy) — należy zawsze przeanalizować problem i zastanowić się, czy wyskakiwanie ma być stosowane, czy też należy wyświetlać ciąg znaków za pomocą jednej z przeznaczonych do tego instrukcji (funkcji). Sekcja z kodem formularza mogłaby mieć też postać zaprezentowaną na listingu 5.3.

Listing 5.3. Alternatywna definicja formularza

```
<?php
```

```
 // Funkcja  wyświetlająca formularz

 function formularz() {

 //definicja  formularza (składnia heredoc)

 $form = <<<EOD

  <form action=&quot;zgloszenie.php&quot;  method=&quot;post&quot;>

  <div>

  Imię:<br />

  <input name=&quot;imie&quot; value=&quot;&quot;  /><br />

  Nazwisko:<br />

  <input name=&quot;nazwisko&quot; value=&quot;&quot;  /><br />

  Zawód:<br />

  <input name=&quot;zawod&quot; value=&quot;&quot;  /><br />

  Adres e-mail:<br />

  <input name=&quot;email&quot; value=&quot;&quot;  /><br />

  <input type=&quot;checkbox&quot;

   name=&quot;mailing&quot; value=&quot;checked&quot;

   />Chcę otrzymywać informacje handlowe<br  /><br />

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

  </div>

  </form>

EOD;

  echo $form;

 }

?>

<body>
```

Kolejna sekcja kodu PHP (na listingu 5.2) pojawia się wewnątrz znacznika body, czyli w części dokumentu HTML przeznaczonej do wyświetlania, i składa się z wywołania funkcji formularz(). Efekt działania tego skryptu jest praktycznie taki sam jak efekt uzyskany po wyświetleniu przykładowego formularza z listingu 5.1.

Znacznik form w przypadku formularza z listingu 5.2, a w przeciwieństwie do formularza z listingu 5.1, zawiera

niepusty atrybut action. W przypadku formularza z listingu 5.2 atrybut ten zawiera adres względny skryptu zgloszenie.php — skrypt ten otrzyma dane z formularza po tym, jak użytkownik kliknie przycisk Wyślij. Listing 5.4 zawiera kod skryptu zgloszenie.php (w tym i dalszych przykładach założono, że w opcjach konfiguracyjnych jest włączone rozpoznawanie skróconych znaczników — short_open_tag = On).

Listing 5.4. Skrypt przyjmujący i przetwarzający dane z formularza

```
<!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>Zgłoszenie</title>
</head>
<body>
<?php
 if (isset($_POST[&quot;submit&quot;])) {
 // Zostaliśmy przekierowani na stronę z formularza
 // - sprawdzamy, czy wszystkie pola zostały wypełnione
  if (empty($_POST[&quot;imie&quot;]) ||
    empty($_POST[&quot;nazwisko&quot;]) ||
    empty($_POST[&quot;zawod&quot;]) || empty($_POST[&quot;email&quot;])) {
   echo &quot;<p style=&quot;color:red&quot;>Musisz wypełnić wszystkie pola!</p>&quot;;
   echo &quot;<p><a href=&quot;formularz.php&quot;>Powrót do formularza</a></p>&quot;;
  } else {
   ?>
   <h3>Dziękujemy za zgłoszenie!</h3>
   <p>Twoje dane:</p>
   <ul>
   <li>Imię: <b><?= trim($_POST[&quot;imie&quot;]); ?></b></li>
   <li>Nazwisko: <b><?= trim($_POST[&quot;nazwisko&quot;]); ?></b></li>
```

```
    <li>Zawód: <b><?=  trim($_POST[&quot;zawod&quot;]); ?></b></li>

    <li>Adres email: <b><?=  trim($_POST[&quot;email&quot;]); ?></b></li>

    <?php

    if (isset($_POST[&quot;mailing&quot;]))

      echo &quot;<li>Chcesz otrzymywać informacje  handlowe.</li>&quot;;

    else

      echo &quot;<li>Nie chcesz otrzymywać informacji  handlowych.</li>&quot;;

    ?>

    </ul>

    <?php

   }

 } else {

   // Jeśli  użytkownik dostał się na tę stronę w sposób inny niż przez formularz

   // zostaje  przekierowany do formularza zgłoszenia

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

 }

?>

</body>

</html>
```

Skrypt z listingu 5.4 zawiera kilka nowości i trzeba go dokładnie przeanalizować. Sekcja kodu PHP zaczyna się wewnątrz ciała strony (znacznik body). Na samym początku za pomocą instrukcji warunkowej sprawdzamy, czy jest ustawiona wartość pola o kluczu submit tablicy $_POST:

```
if (isset($_POST[&quot;submit&quot;])) {

...
```

To, czy ustawiona jest określona wartość zmiennej (w tym przypadku jest to jeden z elementów tablicy asocjacyjnej), sprawdzamy za pomocą funkcji isset(). Tablica $_POST jest jedną z tablic superglobalnych, dostępnych w każdym miejscu skryptu. $_POST zawiera wszystkie dane, które zostały przesłane do aktualnie przetwarzanego skryptu za pomocą metody HTTP POST. Analogicznie, dane przesłane skryptowi (serwerowi) za pomocą metody HTTP GET dostępne są przez tablicę superglobalną $_GET. Gdyby się okazało, że wartość elementu submit tablicy $_POST w naszym skrypcie nie jest ustawiona, wtedy możemy być pewni, że użytkownik dostał się na tę stronę w inny sposób — nie przez formularz (sytuacja opisana w dalszej części punktu).

Jeśli przyjrzymy się skryptowi z listingu 5.2 (formularz), to zauważymy, że element submit tabeli $_POST

odpowiada nazwie (atrybut name) przycisku Wyślij formularza — oznacza to, że jeżeli wartość elementu submit tablicy $_POST jest ustawiona, kliknięty został przycisk Wyślij formularza. Pozostałe elementy formularza powinny być również dostępne w tablicy $_POST pod indeksami odpowiadającymi ich nazwom (wartościom atrybutów name).

Wróćmy do analizy skryptu z listingu 5.4. Jeśli kliknięto przycisk Wyślij formularza (opisywany wyżej warunek jest spełniony), sprawdzamy, czy wszystkie pola formularza zostały zapełnione (wiadomo już, jak odczytać wartość poszczególnych pól formularza), a raczej czy któreś z pól przypadkiem nie pozostało niewypełnione:

```
if (empty($_POST[&amp;quot;imie&amp;quot;]) ||

    empty($_POST[&amp;quot;nazwisko&amp;quot;]) ||

    empty($_POST[&amp;quot;zawod&amp;quot;]) || empty($_POST[&amp;quot;email&amp;quot;])) {

  echo &amp;quot;<p style=&amp;quot;color:red&amp;quot;>Musisz wypełnić wszystkie pola!</p>&amp;quot;;

  echo &amp;quot;<p><a href=&amp;quot;formularz.php&amp;quot;>Powrót do formularza</a></p>&amp;quot;;

} else {

...
```

Instrukcja if zawiera w tym przypadku warunek złożony — wyrażenie logiczne będące alternatywą warunków prostych. Każdy z warunków prostych, np. empty($_POST["imie"]), składa się z wywołania funkcji empty() z odpowiednim elementem tablicy $_POST jako parametrem. Funkcja ta zwraca wartość true, jeżeli zmienna (element tablicy) jest pusta — zmienna istnieje, ale nie przypisano jej wartości, w tym przypadku zmienna zawiera pusty ciąg znaków. Zastosowanie alternatywy (logiczne LUB) powoduje, że jeżeli którekolwiek z pól pozostało puste, skrypt wygeneruje komunikat (wraz z odpowiednim łączem) informujący użytkownika o konieczności wypełnienia formularza. Jeśli wszystkie pola zostały zapełnione, warunek nie jest spełniony, czyli wykonywany jest kod umieszczony po słowie kluczowym else:

```
?>

<h3>Dziękujemy za zgłoszenie!</h3>

<p>Twoje dane:</p>

<ul>

<li>Imię: <b><?= trim($_POST[&amp;quot;imie&amp;quot;]); ?></b></li>

<li>Nazwisko: <b><?= trim($_POST[&amp;quot;nazwisko&amp;quot;]); ?></b></li>

<li>Zawód: <b><?= trim($_POST[&amp;quot;zawod&amp;quot;]); ?></b></li>

<li>Adres email: <b><?= trim($_POST[&amp;quot;email&amp;quot;]); ?></b></li>

<?php

if (isset($_POST[&amp;quot;mailing&amp;quot;]))

  echo &amp;quot;<li>Chcesz otrzymywać informacje handlowe.</li>&amp;quot;;

else
```

```
  echo &quot;<li>Nie chcesz otrzymywać informacji  handlowych.</li>&quot;;

?>

</ul>

<?php
```

Jest to kod HTML (zauważyć można „wyskok” z PHP) wyświetlający dane z formularza, przy czym w miejscach, gdzie dane powinny być wyświetlone, użyte zostały sekcje wyrażeń zawierające wywołania funkcji trim() z odpowiednim elementem tablicy $_POST jako parametrem — funkcja trim() ma za zadanie wyciąć z ciągu znaków wszystkie białe znaki (czyli spacje, tabulatory, znaki końca wiersza) występujące na początku i na końcu ciągu. Dzięki wykorzystaniu funkcji trim() zabezpieczamy się przed ewentualnym nagromadzeniem spacji przed i po napisie. Należy zwrócić uwagę na sekcję kodu PHP dotyczącą obsługi pola wyboru o nazwie mailing. Nazwy pól wyboru wraz z wartościami przesyłane są do skryptu wyznaczonego w atrybucie action formularza tylko wtedy, gdy pola te zostały zaznaczone — stąd zastosowanie instrukcji warunkowej i funkcji isset(). Jeżeli użytkownik zaznaczył pole wyboru, element tablicy $_POST o indeksie mailing został utworzony i jest dostępny.

Wróćmy teraz do przypadku, gdy użytkownik dostał się na stronę zgloszenie.php inaczej niż przez formularz (warunek sprawdzający, czy kliknięto przycisk Wyślij, wygeneruje wartość false). Sytuację taką można obsłużyć na wiele sposobów. Obecnie preferuje się przekierowywanie użytkownika na stronę formularza, z którego dopiero może się dostać do danej strony. W przypadku skryptu z listingu 5.4 problem ten rozwiązany został przez natychmiastowe przekierowanie użytkownika na stronę formularza przy użyciu funkcji header():

```
header(&quot;Location: formularz.php&quot;);
```

Funkcja ta powoduje wysłanie nagłówka HTTP — w przypadku skryptu z listingu 5.4 nagłówek zawiera informację o przekierowaniu na stronę o adresie względnym formularz.php.

Rysunek 5.2 przedstawia wypełniony formularz, natomiast rysunek 5.3 przedstawia wynik działania skryptu zgloszenie.php, do którego zostały przesłane dane formularza.

Rysunek 5.2. Wypełniony formularz

Rysunek 5.3. Przetworzone dane z formularza

Rysunek 5.4 przedstawia wynik działania skryptu zgloszenie.php w przypadku, gdy użytkownik nie wypełnił któregoś z pól formularza.

Rysunek 5.4. Wynik działania skryptu zgloszenie.php w sytuacji, gdy nie zostały wypełnione wszystkie pola formularza

Mechanizm PostBack

Nie jest koniecznością przetwarzanie danych z formularza w oddzielnym skrypcie. Możemy tak przepisać kod

skryptu realizującego formularz, że będzie on w stanie również przetworzyć dane z tego formularza. Innymi słowy, atrybutowi action formularza przypiszemy adres względny bieżącego skryptu i będziemy sprawdzać, czy odwołanie do strony nastąpiło przez formularz, czy też w sposób bezpośredni.

Przekazywanie danych z formularza realizowanego przez skrypt o nazwie np. formularz.php do tego samego skryptu, które jest w stanie również przetworzyć te dane, określane jest mianem PostBack — informacje są „przesyłane z powrotem" do skryptu, który je wygenerował.

Przyjrzyjmy się skryptowi z listingu 5.5 (plik formularz.php).

Listing 5.5. Skrypt realizujący mechanizm PostBack

```
<!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>Formularz</title>

</head>

<?php

 // Funkcja wyświetlająca formularz

 function formularz($imie = &quot;&quot;, $nazwisko = &quot;&quot;, $zawod = &quot;&quot;,

  $email = &quot;&quot;, $mailing_checked = false ) {

  ?>

  <form action=&quot;&quot; method=&quot;post&quot;>

  Imię:<br />

  <input name=&quot;imie&quot; value=&quot;<?= $imie; ?>&quot; /><br />

  Nazwisko:<br />

  <input name=&quot;nazwisko&quot; value=&quot;<?= $nazwisko; ?>&quot; /><br />

  Zawód:<br />

  <input name=&quot;zawod&quot; value=&quot;<?= $zawod; ?>&quot; /><br />

  Adres e-mail:<br />

  <input name=&quot;email&quot; value=&quot;<?= $email; ?>&quot; /><br />

  <input type=&quot;checkbox&quot; name=&quot;mailing&quot;
```

```
    value=&quot;checked&quot;<?php if ($mailing_checked)  echo &quot; checked&quot; ?>
    />Chcę otrzymywać informacje handlowe<br  /><br />
  <input type=&quot;submit&quot; value=&quot;Wyślij&quot;  name=&quot;submit&quot;/>
  </form>
  <?php
 }
?>
<body>
<?php
 if (isset($_POST[&quot;submit&quot;])) {
  // Sprawdzamy,  czy użytkownik zaznaczył pole wyboru i ustawiamy wartość
  // odpowiedniej  zmiennej
  $mailing = false;
  if (isset($_POST[&quot;mailing&quot;]))
   $mailing = true;
  // Wyświetlamy  formularz z wpisanymi wartościami poszczególnych pól
  formularz($_POST[&quot;imie&quot;],  $_POST[&quot;nazwisko&quot;],
   $_POST[&quot;zawod&quot;], $_POST[&quot;email&quot;],  $mailing);
  // Sprawdzamy,  czy wszystkie pola zostały zapełnione
  if (empty($_POST[&quot;imie&quot;]) ||  empty($_POST[&quot;nazwisko&quot;])
     || empty($_POST[&quot;zawod&quot;]) ||  empty($_POST[&quot;email&quot;])) {
   echo &quot;<p style=&quot;color:red&quot;>Musisz  wypełnić wszystkie pola!</p>&quot;;
  } else {
   echo &quot;<p>Dziękujemy za wypełnienie  formularza!</p>&quot;;
  }
 } else {
  // Jeśli  strona ładowana pierwszy raz, wyświetlamy pusty formularz
  formularz();
```

```
 }

?>

</body>

</html>
```

Kod przedstawiony na listingu 5.5 nie jest optymalny, ale realizuje dokładnie to, o co nam chodzi.

Funkcja formularz() jest tym razem bardziej rozbudowana niż w przypadku z listingu 5.2. Funkcja formularz() z listingu 5.5 zawiera listę parametrów, które wykorzystywane są do wypełnienia odpowiednich pól formularza. Parametry te mają wartości domyślne — puste ciągi znaków i wartość false w przypadku parametru, na podstawie którego funkcja generuje pole wyboru domyślnie zaznaczone lub nie. Dokładną analizę tej funkcji warto przeprowadzić w ramach ćwiczenia.

W ciele strony WWW (znacznik body) umieszczony został kod PHP, który najpierw sprawdza — podobnie jak to miało miejsce w przypadku skryptu z listingu 5.4 — czy użytkownik dostał się na stronę przez kliknięcie przycisku Wyślij formularza, czy też w inny sposób. Jeśli użytkownik został przekierowany na tę stronę w wyniku kliknięcia przycisku, najpierw zapamiętywany jest stan pola wyboru (można to zrobić oczywiście prościej) w zmiennej $mailing:

```
$mailing = false;

if (isset($_POST[&amp;quot;mailing&amp;quot;]))

 $mailing = true;
```

Następnie wyświetlany jest formularz zawierający dokładnie takie dane, jakie zostały przesłane za pomocą POST (jako parametr decydujący o zaznaczeniu pola wyboru przekazujemy zmienną $mailing):

```
formularz($_POST[&amp;quot;imie&amp;quot;], $_POST[&amp;quot;nazwisko&amp;quot;],

 $_POST[&amp;quot;zawod&amp;quot;], $_POST[&amp;quot;email&amp;quot;], $mailing);
```

Na koniec następuje sprawdzenie, czy wszystkie pola zostały wypełnione (użytkownik otrzymuje odpowiedni komunikat zwrotny):

```
  if (empty($_POST[&amp;quot;imie&amp;quot;]) || empty($_POST[&amp;quot;nazwisko&amp;quot;])

     || empty($_POST[&amp;quot;zawod&amp;quot;]) || empty($_POST[&amp;quot;email&amp;quot;])) {

    echo &amp;quot;<p style=&amp;quot;color:red&amp;quot;>Musisz wypełnić wszystkie pola!</p>&amp;quot;;

  } else {

    echo &amp;quot;<p>Dziękujemy za wypełnienie formularza!</p>&amp;quot;;

  }
```

Jeśli użytkownik trafił na stronę nie po przekierowaniu z formularza, wywoływana jest funkcja formularz() bez parametrów, co spowoduje wyświetlenie pustego formularza.

Rysunek 5.5 przedstawia wynik działania skryptu z listingu 5.5 po przesłaniu wypełnionego formularza, natomiast rysunek 5.6 przedstawia wynik działania skryptu, gdy formularz nie został poprawnie wypełniony.

Rysunek 5.5. Wynik działania skryptu z listingu 5.5 w przypadku kompletnych danych

Rysunek 5.6. Wynik działania skryptu z listingu 5.5 w przypadku niekompletnych danych

Wiadomo już, jak przetwarzać dane przekazane przez użytkownika za pomocą formularzy HTML. Warto teraz — w ramach ćwiczenia — zaprogramować obsługę innych pól formularzy HTML. W następnych rozdziałach zostanie m.in. pokazane, w jaki sposób można magazynować dane pobrane z formularzy.